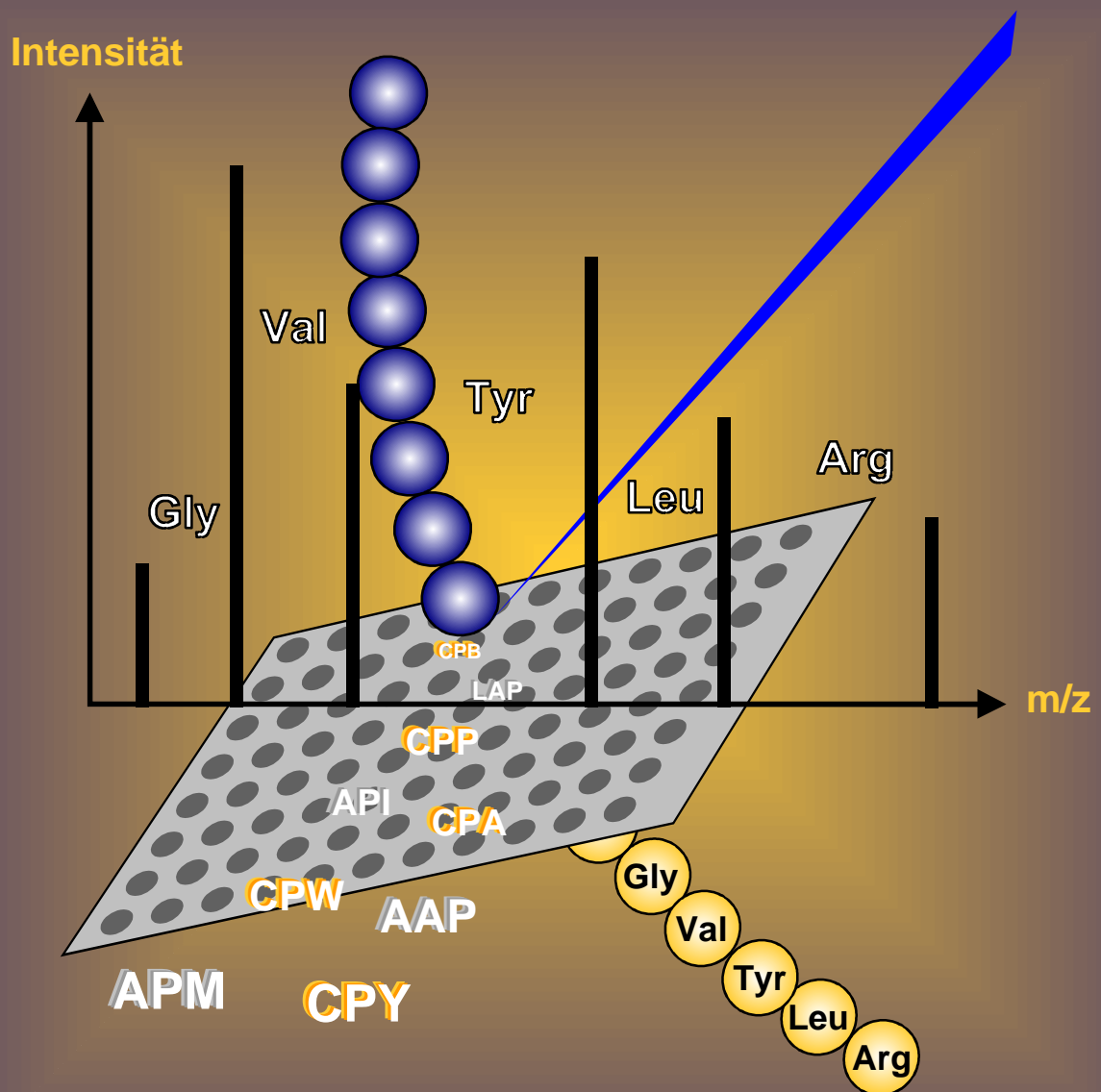


Entwicklung einer *High-Throughput-Sequenzierungsmethode* für die Proteomanalytik



Robert Georg Kratzer

Dissertation

zur Erlangung des Doktorgrades
der Fakultät für Chemie und Pharmazie
der Ludwig-Maximilians-Universität München



Entwicklung einer *High-Throughput*-Sequenzierungsmethode für die Proteomanalytik

Robert Georg Kratzer
aus
Augsburg

München, März 2001

Erklärung

Diese Dissertation wurde im Sinne von §13 Abs. 3 bzw. 4 der Promotionsordnung vom 29. Januar 1998 am Max-Planck-Institut für Biochemie in Martinsried in der Arbeitsgruppe Proteinanalytik von Herrn Dr. habil. Friedrich Lottspeich durchgeführt und von Herrn Prof. Dr. Dieter Oesterhelt betreut.

Ehrenwörtliche Versicherung

Diese Dissertation wurde selbständig, ohne unerlaubte Hilfe erarbeitet.

München, am 06.03.2001

A handwritten signature in black ink, appearing to read 'Robo Kratzer', written over a horizontal line.

Robert Kratzer

Dissertation eingereicht am: 08.03.2001

1. Berichterstatter: Prof. Dr. Dieter Oesterhelt

2. Berichterstatter: PD Dr. Christoph Eckerskorn

Mündliche Prüfung am: 07.05.2001

Inhaltsverzeichnis

Kapitel I: Einleitung	1
1 Proteinidentifizierung und Proteinsequenzierung	2
1.1 Methodische Grundlagen	2
1.2 Methodische Fortschritte	6
2 Aufgabenstellung	9
Kapitel II: Enzymatische Leitersequenzierungen	12
1 Theoretische Grundlagen	14
1.1 Grundlagen der Enzymatischen Sequenzierung	14
1.1.1 Exopeptidasen	14
1.1.2 Möglichkeiten zur Verfolgung der Aminosäureabspaltung	19
1.1.3 Kinetische Steuerung der Aminosäureabspaltung	22
1.1.4 Peptidase-Kombinationen	26
1.2 Grundlagen der Leiterpeptid-Massenanalyse mittels MALDI-TOF-MS	28
1.2.1 Prinzip der Massenanalyse mit MALDI-TOF	28
1.2.2 Spezielle Aspekte beim enzymatischen Leitersequenzieren	35
1.2.2.1 Spektreninterpretation, Massengenauigkeit und isobare Aminosäuren	35
1.2.2.2 Der Einfluss von Puffersalzen	36
1.2.2.3 Analytisch nutzbarer Bereich des Massenspektrums	36
1.3 Praktische Ausführungsformen der Leitersequenzierung	40
1.3.1 Spaltung in Lösung, <i>on-target</i> oder auf der Membran	40
1.3.2 Zeit- und Konzentrationsabhängige Spaltungen	44
1.4 Weitere Aspekte der enzymatischen Leitersequenzierung	51
1.4.1 Proteinspaltung	51
1.4.1.1 Probleme bei der Sequenzierung intakter Proteine	51
1.4.1.2 Sequenzierung terminal modifizierter Proteine	52
1.4.1.3 Wahl der Endopeptidase für die Proteinspaltung	53
1.4.2 HPLC-Trennung der Proteinfragmente	56
1.4.2.1 Peaküberlappung der Proteinfragmente im Chromatogramm	56
1.4.2.2 Sequenzierbarer Anteil des Proteins	60
1.4.3 Verbesserungsmöglichkeiten der Methode durch Derivatisierungen	61

2	Experimenteller Teil	66
2.1	Verwendete Proteine, Peptide, Enzyme und Chemikalien	66
2.1.1	Exopeptidasen.....	66
2.1.2	Endopeptidasen.....	68
2.1.3	Synthetische Peptide für die Sequenzierung.....	68
2.1.4	Proteine für die Sequenzierung.....	71
2.1.5	Chemikalien.....	72
2.2	Verwendete Geräte.....	74
2.2.1	HPLC.....	74
2.2.2	Massenspektrometer	75
2.2.3	Sonstige Geräte und Hilfsmittel.....	76
2.3	Vorbereitung der Proteinproben für die Sequenzierung.....	77
2.3.1	Reduktion und Alkylierung	77
2.3.2	Endopeptidase Spaltungen.....	77
2.3.3	HPLC-Trennungen der proteolytischen Fragmente	79
2.4	MALDI-Probenpräparationen	80
2.4.1	Matrixlösungen.....	80
2.4.2	Standard Probenpräparationen.....	80
2.4.3	Präparationen proteolytischer Fragmente der Endopeptidasespaltungen.....	81
2.5	Enzymatische Leitersequenzierung.....	82
2.5.1	Präparationen für die Sequenzierung	82
2.5.1.1	Sequenzierung in Lösung.....	82
2.5.1.2	Sequenzierung auf dem MALDI-Target (on-target)	83
2.5.1.3	Sequenzierung auf der PVDF-Membranen	83
2.5.2	Anwendungsformen der Exopeptidasen – Einzelenzyme und Enzymkombinationen.....	84
2.6	MALDI-TOF-MS Messungen.....	90
2.6.1	Parameter der UV-MALDI-MS Messung von Peptiden.....	90
2.6.2	Parameter der UV-MALDI-MS Messung von Proteinen	91
2.6.3	Parameter der IR-MALDI-MS Messungen von Peptiden.....	92
2.6.4	Kalibrierung.....	92
2.6.5	Datenakquisition und Datenauswertung	93
2.7	Derivatisierungen	94
2.7.1	Peptidderivatisierung mit Säurechloriden und Succinimidylcarbonaten	94
2.7.2	Peptidderivatisierung mit Isothiocyanaten.....	95
2.7.3	Peptidderivatisierung mit 6-Aminochinolyl-N-hydroxysuccinimid	95
2.7.4	Peptidderivatisierung durch Acetylierung	95
2.7.5	HPLC-Reinigung der Peptidderivate	96

3 Ergebnisse	97
3.1 Synthetische Peptide	97
3.1.1 Sequenzen.....	97
3.1.2 Abbaulängen, Sequenzlängen und Sequenzabdeckung	98
3.1.3 Massengenauigkeit	101
3.1.4 Empfindlichkeit und Dynamik.....	102
3.1.5 pH-Bedingungen.....	110
3.2 Proteolytische Spaltpeptide	113
3.2.1 Sequenzen.....	114
3.2.1.1 Sequenzen aus Spaltungen mit Endoproteinase LysC	115
3.2.1.2 Sequenzen aus Spaltungen mit Endoproteinase GluC (Phosphatpuffer)	118
3.2.1.3 Sequenzen aus Spaltungen mit Trypsin	121
3.2.1.4 Sequenzen aus Spaltungen mit Chymotrypsin.....	123
3.2.1.5 Sequenzen aus Spaltungen mit Endoproteinase AspN.....	127
3.2.2 Grafische Darstellung der Sequenzierungsergebnisse – Sequenzalignments.....	127
3.2.3 Statistische Auswertung der Sequenzierungsergebnisse	134
3.2.3.1 Definition des sequenzierbaren Anteils	134
3.2.3.2 Sequenzabdeckungen der untersuchten Proteine	136
3.2.3.3 Sequenzlängenverteilung in Abhängigkeit der Endopeptidasespaltung.....	140
3.2.3.4 Sequenzlängenverteilung in Abhängigkeit der Exopeptidasespaltung.....	146
3.3 Derivatisierte Peptide	153
3.3.1 Derivatisierung zur Unterscheidung Glutamin / Lysin	153
3.3.2 Derivatisierung zur Sequenzierung kleiner Peptide.....	155
3.3.2.1 Überlegungen zur Auswahl von Derivatisierungsreagenzien	155
3.3.2.2 Selektivitäten und Produktspektren	158
3.3.2.3 Sequenzierung der Peptidderivate.....	161
3.3.2.4 Eigenschaften der Peptidderivate.....	165
3.3.3 Bromhaltige Peptidderivate	167
3.3.4 Erforderliche Peptidmengen für die Derivatisierungen	169
3.4 Post-translationale Modifikationen	171
3.5 Einfluss der MALDI-Matrix und Suppressionseffekte	177
3.6 Sequenzierung auf der PVDF-Membran.....	183
4 Diskussion und Bewertung.....	187
4.1 Experimentelle Bedingungen	187
4.1.1 Optimale Spaltungstemperaturen und Spaltungszeiten.....	187
4.1.2 Enzyme, Enzymkombinationen und Enzymmengen	189
4.1.3 pH-Bedingungen.....	193
4.1.4 Praktische Sequenzierungsstrategien	196
4.2 Empfindlichkeit und Dynamik	198

4.3	Spektreninterpretation	203
4.3.1	Oxidationen	203
4.3.2	Metastabile Fragmentierungen	207
4.3.3	Unterscheidung isobarer Aminosäuren.....	208
4.3.4	Parallele Sequenzierungen mehrerer Peptide.....	210
4.4	Sequenzierung modifizierter Peptide	213
4.4.1	Synthetische Modifikationen	213
4.4.2	Post-translationale Modifikationen.....	218
4.5	Kriterien für Sequenzabbrüche.....	219
4.6	Sequenzlängen und Anwendungsbereich	225

Kapitel III: Automatisierung **232**

1	Anforderungen an die Methode.....	234
2	Experimenteller Teil	238
2.1	Verwendete Proteine, Peptide, Enzyme und Chemikalien	238
2.2	Verwendete Geräte.....	238
2.2.1	Automatisches Pipettiersystem	238
2.2.2	Massenspektrometer	240
2.3	Verwendete Softwarekomponenten	241
2.3.1	Automatisches Pipettiersystem	241
2.3.2	Automatische MALDI-MS Messungen.....	241
2.3.3	Zusätzliche Softwarekomponenten.....	241
3	Ergebnisse	242
3.1	Die Sequenziermethode „MALDI-LS 1.42s“ für den MultiPROBE II	242
3.1.1	Allgemeines zur Sequenziermethode „MALDI-LS 1.42s“	242
3.1.2	Arbeitsschritte in der Sequenziermethode „MALDI-LS 1.42s“	243
3.1.3	Einzelaspekte der Methode „MALDI-LS 1.42s“	246
3.1.3.1	Belegung des MultiPROBE II mit unterschiedlichen Racks.....	246
3.1.3.2	Vorbereitung der Lösungen – Verteilungsschritte (I)	248
3.1.3.3	Probenverschleppung und sonstige Kontaminationen	248
3.1.3.4	Verwendung der Pufferschritte 1 bis 4.....	249
3.1.3.5	Kalibrierung	250
3.1.3.6	Parameter der Flüssigkeitsaufnahme und –abgabe	250
3.1.3.7	Spezielle Laufzeitvariablen im Programmablauf.....	252
3.1.4	Dateneingabe	254
3.1.4.1	Die Datei „FLAG PARAMETERS.TXT“	255

3.1.4.2	Die Dateien „STOCK SOLUTION SOURCE.TXT“ und „STOCK SOLUTION DESTINATION.TXT“	256
3.1.4.3	Die Datei „SAMPLES.TXT“ – Gemeinsame Dateneingabe für MultiPROBE II und AutoXecute	258
3.1.4.4	Datenaustausch zwischen MultiPROBE II und MALDI-MS	261
3.2	Automatische MALDI-MS Messungen	261
3.2.1.1	Optimierung der Probenpräparation.....	263
3.2.1.2	Optimierung des Abtastungsrasters der Spots.....	264
3.2.1.3	Kalibrierung des MALDI-Targets für die AutoXecute.....	266
3.2.1.4	Einzelaspekte der optimierten AutoXecute Methode.....	267
3.2.1.5	Kalibrierung	270
3.2.1.6	Der Einfluss von Salzen.....	271
3.3	Ergebnisse für unterschiedliche Stufen der Automatisierung	274
3.3.1	Automatisierungsstufe 1	274
3.3.2	Automatisierungsstufe 2	278
3.3.3	Automatisierungsstufe 3	280
3.3.4	Automatisierungsstufe 4	282
4	Diskussion und Bewertung	287
4.1	Reaktionszeiten bei der Sequenzierung.....	287
4.2	Technische Verbesserungen im Teilbereich der automatischen Probenpräparation	288
4.3	Kompatibilität der Probenträgerformate zwischen Pipettierrobot und MALDI-MS	289
4.4	Automatische MALDI-MS Messungen	289
4.5	Aussagekraft negativer Resultate	292

Kapitel IV: Zusammenfassung und Ausblick

294

Anhang

300

A.	Abkürzungsverzeichnis.....	302
B.	Nomenklatur der Peptidmodifikationen	304
C.	Abkürzungen der Peptidasen.....	305
D.	Aminosäuren / Aminosäurereste.....	306
E.	Verzeichnis der Dateien auf beiliegender CD-ROM	308
F.	Literaturverzeichnis.....	310

Kapitel I:

Einleitung

1 Proteinidentifizierung und Proteinsequenzierung

1.1 Methodische Grundlagen

Die Identifizierung von Proteinen über die Primärstruktur nimmt in der Biochemie seit je her eine wichtige Position ein. Als durch den Edman-Abbau [Edman, 1950] zur Mitte des 20. Jahrhunderts die ersten effizienten Sequenzierungstechniken für Proteine entwickelt wurden, musste man den Begriff der „Identifizierung“ oft noch mit einer vollständigen Sequenzierung des Proteins gleichsetzen. Daher kommt solchen Methoden, die eine direkte Information der Aminosäuresequenz eines Proteins liefern, in der Primärstrukturanalyse eine fundamentale Bedeutung zu. Mit dem Aufkommen der Proteomforschung [Wilkins 1995, Wilkins_1 1996, Lottspeich 1999] Mitte der 90er Jahre gewannen Methoden zur Proteinidentifizierung noch weiter an Bedeutung. Auch in der Proteomanalytik finden dabei klassische Sequenzierungsmethoden, insbesondere der Edman-Abbau, noch starke Anwendung. Doch die in der Proteomanalytik oft extrem hohe Zahl zu identifizierender Proteinproben verlangt nach immer schnelleren und effizienteren Identifizierungsmethoden.

Ein wichtiges Instrument für die Proteomanalyse stellt dabei sicherlich die Proteinidentifizierung über Homologiesuchen in Protein- und DNA-Datenbanken dar [Mewes 1994]. Informationen über bekannte Protein- und DNA-Sequenzen werden seit Anfang 1980 in Datenbanken gespeichert. Bereits in den Anfängen der Proteomforschung 1995 betrug allein die Zahl der Einträge in die PIR Proteindatendatenbank ca. 70000 Sequenzen. Mit dem parallelen Fortschritt der Molekularbiologie in den letzten beiden Jahrzehnten, insbesondere der einfachen Möglichkeit aus genomischen Daten die Primärstruktur der codierten Proteine zu erhalten, ergab sich für die rasche Proteinidentifizierung somit ein großes Potential. Neben stetig wachsenden Proteinsequenzdatenbanken, wie z.B. SWISS-PROT oder NBRF-PIR kam es in den letzten Jahren, getragen durch die Vielzahl der großen Genomprojekte, zu einem exponentiellen Anstieg der Einträge in Nucleotiddatenbanken, wie z.B. GenBank oder EMBL. Ebenso positiv wirkten sich in diesem Zusammenhang Entwicklungen in der Bioinformatik als einem interdisziplinären Bereich der Biochemie aus, sowie technische Weiterentwicklungen der Rechnerleistung im Computerbereich. Effiziente Suchalgorithmen wie FASTA oder BLAST ermöglichen es heute auf Hochleistungsrechnern, die kontinuierlich wachsende Zahl von Datenbankeneinträgen in Millisekunden nach kurzen Partialsequenzen eines Proteins zu durchsuchen. Somit kann ein Protein unter Umständen mit extrem geringen Aufwand und gleichzeitig einem hohen Maß an Sicherheit identifiziert werden. Durch das umfassende und stetig wachsende Sequenzdatenmaterial ist heutzutage eine wichtige

Voraussetzung für die schnelle Proteinidentifizierung – nicht nur in der Proteomanalyse – geschaffen.

Dennoch stellt eine Datenbankrecherche im Rahmen einer Proteinidentifizierung keinen universellen Lösungsansatz dar. In vielen Fällen liefert eine Homologiesuche entweder kein eindeutiges Resultat oder überhaupt keinen Treffer. Daher besteht auch weiterhin ein starkes Interesse und ein Bedarf an schnellen Methoden zur *de novo* Sequenzierung. Vor allem aber unter dem Aspekt der Funktionsanalyse von Proteinen ist in den letzten Jahren die Entwicklung von Sequenzierungsmethoden, die eine einfache Lokalisierung post-translationaler Modifikationen erlauben, in den Mittelpunkt des Interesses gerückt [Meyer 1994, Lottspeich 1999]. Neben der Steigerung der Empfindlichkeit und des Probendurchsatzes ist dies wohl eines der entscheidenden Argumente für Neu- und Weiterentwicklungen von Methoden auf dem Gebiet der Primärstrukturanalytik. Auf der anderen Seite verlangt natürlich auch ein Weg der Proteinidentifizierung über Datenbankrecherche nach entsprechenden Sequenzierungstechniken, die einen hohen Probendurchsatz erlauben. Für probenintensive Untersuchungen wie sie z.B. in der Proteomanalytik auftreten, müssen dabei viele Partialsequenzen ausreichender Länge in möglichst kurzer Zeit erzeugt werden. Die Edman-Sequenzierung kann diesem Anspruch heute nicht mehr gerecht werden.

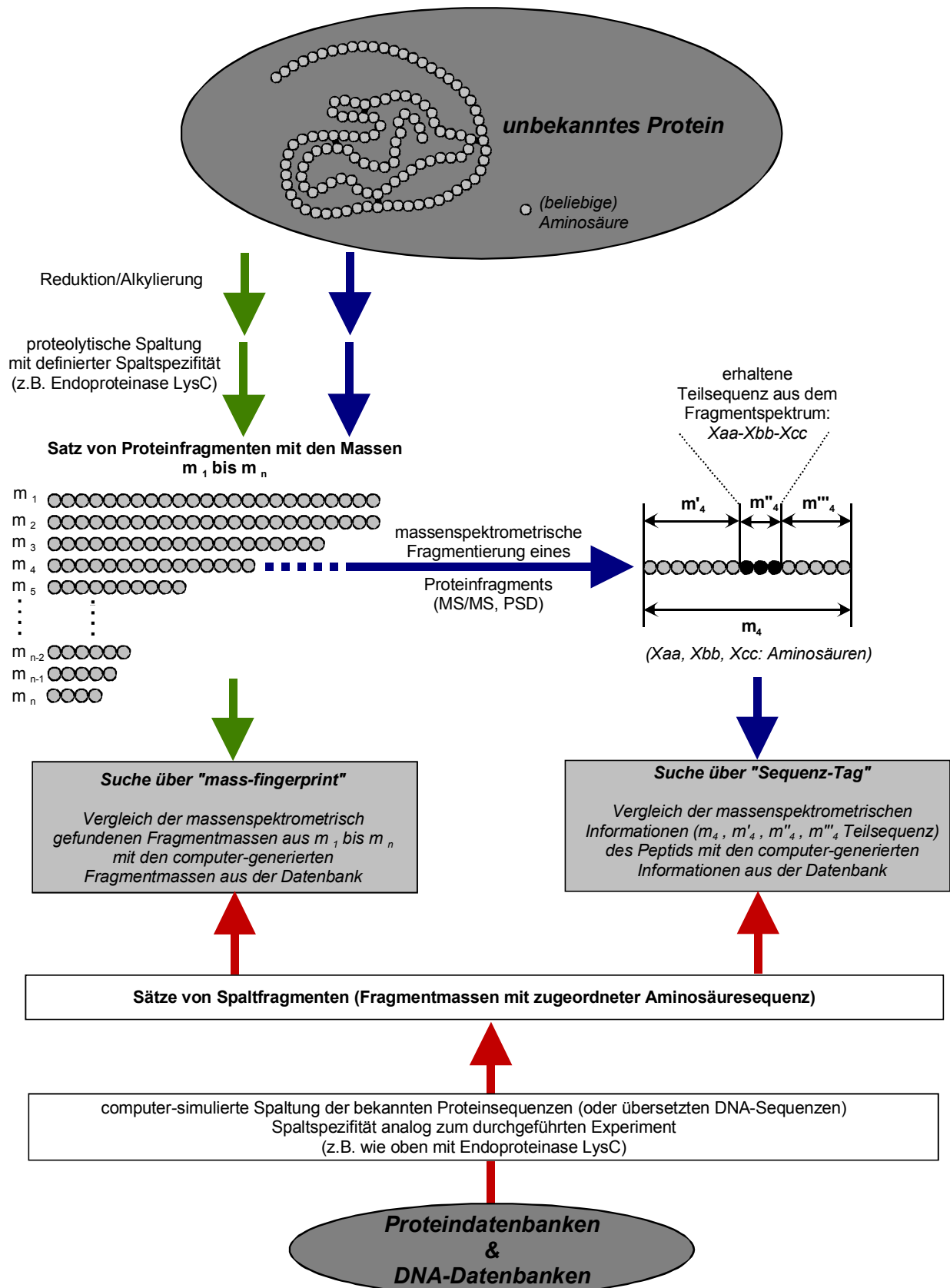
Gerade moderne massenspektrometrische Methoden zeigen ein enormes Potential um allen oben genannten Forderungen bei der schnellen Proteinanalyse gerecht zu werden, sowohl bei der Identifizierung bekannter Proteine über Suche in Sequenzdatenbanken, als auch bei der Primärstrukturaufklärung unbekannter Proteine. Dabei haben vor allem die Elektrospray-Ionisations-Massenspektrometrie (ESI-MS) [Biemann 1987] als auch die Matrix-unterstützte Laserdesorptions/Ionisations-Massenspektrometrie (MALDI-MS) [Karas 1988] mittlerweile eine herausragende Stellung in der Proteinanalytik eingenommen [Shevchenko 1996, Jensen 1998, Süßmuth 1999, Yates 2000]. Beide Verfahren erlauben die Sequenzanalyse von proteolytisch oder chemisch erzeugten Proteinfragmenten über deren anschließende Fragmentierung im Massenspektrometer [Spengler 1992, Kaufmann 1993, Yates 1996]. Solche rein massenspektrometrischen Sequenzierungen wurden gerade zum Ende des letzten Jahrzehnts stark verbessert und werden heute in der Proteinanalyse überwiegend zur Generierung kurzer Sequenztags und anschließender Homologiesuche in Datenbanken genutzt [Mann 1993, Wilkins_2 1996]. *De novo* Sequenzierungen sind zwar mit den zur Verfügung stehenden Methoden der Massenspektrometrie theoretisch auch schon heute sehr empfindlich möglich, allerdings stellt die Interpretation der zum Teil sehr komplizierten Fragmentierungsmuster von Peptiden und Proteinen noch eine entscheidende Limitierung für die routinemäßige Anwendung dar. Eine Interpretation der Fragmentspektren „per Hand“ ist meist sehr zeitaufwendig und auch häufig nur für Peptide kürzerer bis mittlerer Sequenzlänge

(bis etwa 20 Aminosäuren) überhaupt möglich. Computeralgorithmen, die bei der Strukturaufklärung eine wichtige Hilfe darstellen, werden zwar ständig weiterentwickelt, sie nehmen dem Biochemiker jedoch die Interpretationsarbeit nicht vollständig ab. Da der Arbeitsaufwand somit in vielen Fällen beträchtlich ist und auch die letztlich gefundene Aminosäuresequenz nicht immer eindeutig ist, lässt sich im Bereich der massenspektrometrischen Sequenzierung unbekannter Proteine gegenüber der Edman-Sequenzierung oft kein entscheidender Vorteil erkennen.

Bei der Proteinidentifizierung über Datenbank-basierende Sequenzsuchen gewinnen massenspektrometrische Techniken dennoch eine immer größere Bedeutung und haben andere Identifizierungsmethoden bereits stark verdrängt. Zur Zeit sind die massenspektrometrische Analyse proteolytisch oder chemisch erzeugter, interner Proteinfragmente, sogenannte *mass fingerprints*, und „tag“-Sequenzierungen die populärsten Techniken [Mann 1993, Jensen 1996, Perkins 1999]. Die Prinzipien dieser beiden wichtigen Techniken ist in Abbildung 1 zusammengefasst. Bei der Proteinidentifizierung über einen *mass fingerprint* wird mittels massenspektrometrischer Analyse das komplette Gemisch der internen Fragmente nach der Proteinspaltung untersucht. Die Angabe der eingesetzten Spaltspezifität und der gefundenen Fragmentmassen erlaubt einen anschließenden Vergleich mit den bekannten Sequenzen aus Datenbanken (Proteinsequenzdatenbanken oder übersetzte DNA-Datenbanken). Der Erfolg einer Suche und die Eindeutigkeit eines möglichen Treffers hängen dabei wesentlich von der Genauigkeit der Massenbestimmung und der Zahl der für Datenbanksuche eingegebenen Fragmentmassen ab. Bei Identifizierung über einen *Sequenz-tag* wird lediglich eines der zuvor erzeugten Spaltpeptide für die Identifizierung benötigt. Liefert die massenspektrometrische Fragmentierung eines solchen Spaltpeptids eine kurze Teilsequenz, so ist mit der Summe aller gewonnenen Informationen über dieses eine Spaltpeptid (siehe Abbildung 1) eine Identifizierung durch Datenbankvergleich möglich. Soll eine Proteinidentifizierung allein über Fragmentierungsspektren von internen Proteinfragmenten erfolgen, so existiert z.B. mit dem SEQUEST-Algorithmus [Ducret 1998], bereits eine Möglichkeit die langwierige, manuelle Interpretation des Zerfallsspektrums zu umgehen. Prinzipiell haben Methoden, die zur Analyse auf einem solchen Fingerabdruck des Proteins basieren, eine Reihe entscheidender Vorteile. Neben dem geringen Arbeits- und Kostenaufwand ist auch meist eine weitgehend vollständige Automatisierung möglich.

Bei den im vorangegangenen Abschnitt beschriebenen Methoden der Proteinidentifizierung besteht allerdings auch nicht zu unterschätzender Nachteil: Erhält man bei der Datenbankanalyse des Fingerabdrucks nämlich kein eindeutiges Ergebnis oder sind weder das Protein, noch seine Nucleotidsequenz als Eintrag in einer Datenbank enthalten, so muss zusätzlich eine direkte Analyse der Primärsequenz durchgeführt werden.

Abbildung 1: Prinzip der Proteinidentifizierung über *mass-fingerprint* und massenspektrometrisch erzeugten *Sequenz-tag*



Dabei muss dann wieder auf eine der oben genannten Sequenzierungsmethoden (z.B. Edman-Sequenzierung) zurückgegriffen werden. Eine nicht eindeutige Identifizierung, selbst bei einem vorhandenen Datenbankeintrag für das untersuchte Protein, kann mehrere Ursachen haben und daher häufiger auftreten. Vor allem wenn die Zahl der im Massenspektrum gefundenen Fragmente nicht groß genug ist, kann es zu mehrdeutigen oder falschen Aussagen kommen. Des weiteren setzt eine sichere Identifizierung eine extrem hohe Massengenauigkeit voraus, wie sie nur durch interne Kalibrierung erhalten werden kann. Diese Art der Kalibrierung ist jedoch bei *mass fingerprints* praktisch nur möglich, wenn autoproteolytisch entstandene Enzymfragmente mit bekannter Sequenz (und damit auch bekannter Masse) im Massenspektrum auftreten, wie z.B. bei Spaltungen mit Trypsin. Bei interner Kalibrierung können jedoch sekundäre Probleme entstehen, da eine Signalunterdrückung der Fragmente des untersuchten Proteins durch die internen Kalibranden stattfinden kann. Schließlich können auch unvollständige Spaltungen, Fehlsplaltungen durch proteolytische Fremdaktivität im verwendeten Enzym und unspezifische, chemische Modifikationen des Proteins oder eingeschleppte Verunreinigungen während der Aufarbeitung – insbesondere bei geringen Probenmengen – die Identifizierung deutlich erschweren. Zeit- und Arbeitsaufwand können also auch hier ansteigen und sich summieren.

1.2 Methodische Fortschritte

Unter den bisher diskutierten Gesichtspunkten lässt sich die Eignung einer Methode zur Proteinidentifizierung also prinzipiell unter den folgenden zwei Aspekten beurteilen:

- ❖ Analyse eines unbekannten Proteins oder eines Proteins aus unbekanntem Genom
- ❖ Identifizierung eines bekannten Proteins oder eines Proteins aus bekanntem Genom

Der erste und auch kompliziertere Fall erfordert mit den bisher beschriebenen Methoden eine zeitraubende *de novo* Sequenzierung, während im anderen Fall zwei Wege zur Identifizierung beschritten werden können. Erstens die Datenbanksuche nach kurzen Sequenzabschnitten oder zweitens die rasche aber nicht immer erfolgreiche Identifizierung über Mustervergleiche interner Proteinfragmente oder von Fragmentierungsspektren gegen eine Datenbank.

Im vorangegangenen Abschnitt wurde das Potential, aber auch die Probleme und Nachteile heute verfügbarer Methoden zur Proteinanalyse offensichtlich. Gerade im Hinblick auf die Proteomforschung, ist die Notwendigkeit einer Entwicklung neuer, hochparalleler Sequenzierungsstrategien zur Ergänzung bestehender Techniken evident. Eine ideale Methode sollte die beiden oben herausgestellten Aspekte der Proteinidentifizierung gleichzeitig

berücksichtigen. Neben einer schnellen *de novo* Sequenzierung sollte die Methode also auch bei der Generierung von Sequenztags mit anderen Methoden konkurrieren können. Weitere Forderungen an eine neue Methode beinhalten hohe Empfindlichkeit bis in den Femtomol-Bereich, Automatisierbarkeit und die Möglichkeit, bei der Sequenzanalyse direkt Informationen über post-translationale Modifikationen des Proteins zu erhalten.

Erwähnenswert sind im Hinblick auf die geforderten Eigenschaften Weiterentwicklungen des Edman-Abbaus für die N-terminale Sequenzanalyse und des Schlack-Kumpf-Abbaus für die C-terminale Sequenzanalyse [Schlack 1926], sowie Sequenzierungen basierend auf der Anwendung von Exopeptidasen. Aber auch bei der Sequenzierung über rein massenspektrometrische Techniken wurden entscheidende Verbesserungen eingeführt.

Für die klassischen Sequenzierungstechniken nach Edman und Schlack-Kumpf basieren die erwähnten Optimierungen zum großen Teil auf einer Kopplung mit der Massenspektrometrie. Bei der Variation des Edman-Abbaus wird die Sequenzanalyse nicht mehr wie üblich durch Detektion der sukzessiv freigesetzten Aminosäuren (beziehungsweise deren Derivate). Vielmehr erfolgt jeder Abbauzyklus zu einem geringen Prozentsatz durch partielle N-terminale Blockierung unvollständig und nach der gewünschten Anzahl an Abbauschritten erfolgt eine Analyse des verbleibenden Satzes an N-terminal blockierten Peptiden [Chait 1993]. Diese Technik wurde als sogenannte Leitersequenzierung oder als *ladder sequencing* bezeichnet. Man kann die Aminosäuresequenz aus dem Massenspektrum durch Differenzbildung der konsekutiv kleiner werdenden Peptidmassen einfach ablesen. Die im klassischen Edman-Abbau übliche Zykluszeit von ca. 30min kann dadurch, dass der Abbau hier nicht quantitativ sein muss, auf ca. 5min verkürzt werden. Leitersequenzierungen dieser Art lassen sich nicht nur im Zusammenhang mit der Edman-Chemie realisieren. Eine Reihe anderer Abbaumechanismen lassen sich in analoger Weise ebenfalls zu Leitersequenzierung einsetzen. Als Beispiele solcher Abbaumechanismen sind der modifizierte Schlack-Kumpf-Abbau [Boyd 1992] oder partielle saure Hydrolysen in der Gasphase mit perfluorierten Carbonsäuren [Tsugita 1992, Takamoto 1995] zu nennen. Insbesondere hervorheben lassen sich jedoch rein enzymatische [Caprioli 1986, Thiede 1995, Patterson 1995], sowie gemischte chemisch-enzymatische Ansätze [Thiede 1997].

Auch bei der Sequenzierung via massenspektrometrischer Fragmentierung wurden in den letzten drei Jahren entscheidende Verbesserungen erzielt. Hier zeigte sich ein erfolgreicher Ansatz vor allen in der gezielten Derivatisierung der zu fragmentierenden Peptide. Durch Einführung entsprechender, funktioneller Gruppen in das Peptid – meist N-terminal – gelang es in vielen Fällen das Fragmentierungsverhalten soweit zu beeinflussen, dass sehr einfach zu interpretierende Fragmentspektren resultierten [Bartlett-Jones 1994, Gu 1997]. Die

Derivatisierung tryptischer Peptide mittels cyclischem Sulfobenzoesäureanhydrid [Keough 1999] liefert dabei besonders gute Resultate.

Die enzymatische Sequenzierung mittels Exopeptidasen stellt bezüglich den der Sequenzierung zugrundeliegenden (bio)chemischen Prozessen gegenüber den bisher besprochenen Techniken zunächst eine völlig neu Alternative dar. Die Idee, das Potential von Exopeptidasen für die N- und auch C-terminale Proteinsequenzierung zu nutzen, wurde schon in den 70er Jahren erkannt und in der Literatur beschrieben [Light 1972, Ambler 1972, Tschesche 1977, Hayashi 1977]. Bei den ersten Sequenzierungsversuchen mittels Exopeptidasen verfolgte man die Kinetik der Abspaltung anhand der einzelnen, freigesetzten Aminosäuren direkt. Dazu wurden zu verschiedenen Zeitpunkten des Abbaus die quantitativen Verhältnisse der Aminosäuren untersucht. Als mögliche Analysemethoden für die abgespaltenen Aminosäuren kamen dabei verschiedene chromatographische Trennmethoden wie beispielsweise Ionenaustausch- oder Dünnschichtchromatographie gekoppelt mit anschließender Fluoreszenzdetektion zur Anwendung. Diese mittlerweile „historischen“ Methoden der enzymatischen Sequenzierung konnten jedoch bezüglich der Analysengeschwindigkeit und der Empfindlichkeit gegenüber anderen Sequenzierungsmethoden ihrer Zeit keine Vorteile bieten. Dieses Problem lag aus heutiger Sicht sicherlich an der notwendigen, aber sehr zeitaufwendigen quantitativen Aminosäureanalytik zur Verfolgung des Abbaus.

Die Entwicklung massenspektrometrischer Methoden für die Peptid und Proteinanalytik rückte aber auch die enzymatische Sequenzierung wieder in den Blickpunkt des Interesses. Einerseits bot sich eine zusätzliche Möglichkeit für eine empfindlichere Detektion der abgespaltenen Aminosäuren, andererseits bestand nun zusätzlich, wie beim oben beschriebenen modifizierten Edman-Abbau, die Alternative einer Leitersequenzierung durch die beim enzymatischen Abbau als fassbare Zwischenprodukte auftretenden, sukzessiv verkürzten Peptide. Damit konnte der zeitaufwendige Schritt der Aminosäuretrennung und –quantifizierung entfallen. Für die Detektion der Leiterpeptide kamen Mitte der 80er Jahre zunächst die *Fast Atom Bombardent Massenspektrometrie* (FAB-MS) [Caprioli 1986] und die Plasma-Desorptionsmassenspektrometrie [Klarskov 1992] zum Einsatz. In Kombination mit der MALDI-MS [Thiede 1995, Patterson 1995] verspricht die enzymatische Leitersequenzierung ein enormes Potential für die schnelle *de novo* und *tag*-Sequenzierung und somit auch für die *high throughput*-Analytik im Bereich der Proteomforschung. Bei Realisierung einer geeigneten Automatisierung für die enzymatische Leitersequenzierung könnte diese Methodik einen großen Beitrag leisten, den gegenwärtigen Ansprüchen bei Proteinsequenzierung und Proteinidentifizierung Rechnung zu tragen.

2 Aufgabenstellung

Diese Arbeit beschäftigt sich mit der methodischen Weiterentwicklung und Evaluierung der enzymatischen Leitersequenzierung gekoppelt mit der MALDI-Flugzeit-Massenspektrometrie (MALDI-TOF-MS). Dabei wurde die Ausarbeitung einer Methodik unter zwei Gesichtspunkten betrieben:

- ❖ Optimierung der Sequenzierung und MS-Analytik
- ❖ Automatisierung der Analytik

Die Bearbeitung von des ersten Punktes erfolgte ausgehend von den maßgeblichen Anforderungen an eine zeitgemäße Sequenzierungsmethode, welche nachfolgend kurz zusammengefasst sind:

- ❖ gleichermaßen gute Eignung für die *de novo* oder *tag*-Sequenzierung
- ❖ schnell und für hohen Probendurchsatz geeignet
- ❖ hohe Empfindlichkeit bis in den Femtomol-Bereich
- ❖ hohe Dynamik bei unbekannten und stark variierenden Probenmengen
- ❖ einfache Lokalisierung und Identifizierung post-translationaler Modifikationen

Erste Versuche zur Leitersequenzierung mittels Exopeptidasen und MALDI-TOF-MS haben gezeigt, dass diese Technik den obigen Anforderungen durchaus gerecht werden könnte [Patterson 1995]. Die in der Literatur gezeigten Beispiele beschränkten sich jedoch in der Vergangenheit zumeist auf Versuche mit einigen wenigen, ausgewählten synthetischen Peptiden [Holland 1996, Doucette 1999]. Seltener wurden einzelne Peptide aus proteolytischen Spaltungen von Proteinen getestet. Was jedoch letztlich vollständig fehlt, sind systematische Untersuchungen und Daten, die eine prinzipielle Eignung der Leitersequenzierung in der *de novo* Sequenzierung von Proteinen beinhalten. Solche Untersuchungen verlangen eine ganzheitliche Betrachtung der Arbeitsschritte vom gereinigten, zu untersuchenden Protein über die eigentlichen Sequenzierung bis hin zur Massenspektrometrie. Unter dieser globalen Betrachtungsweise wurde in dieser Arbeit neben Basisversuchen mit einer repräsentativen Anzahl synthetischer Peptide vor allem komplette proteolytische Spaltungen mehrerer verschiedener Proteine mittels Leitersequenzierung untersucht. Hierbei wurde zusätzlich ein möglicher Einfluss verschiedener Endopeptidasen auf den Sequenzierungserfolg studiert.

Während in der Literatur bisher überwiegend C-terminale Leitersequenzierungen mit Carboxypeptidasen beschrieben sind [Thiede 1995, Patterson 1995], fehlen entsprechend

gründliche Evaluierungen für die N-terminale Sequenzierung mit Aminopeptidasen. Einzelbeispiele finden sich in [Holland 1996] und [Doucette 1999]. Die Sequenzierungsstudien wurden daher in dieser Arbeit gleichermaßen für beide Termini unternommen, wobei eine Reihe kommerziell erhältlicher Carboxy- und Aminopeptidasen getestet wurden.

Bei den Untersuchungen lag der Schwerpunkt eingangs auf Sequenzierungsversuchen mit nur einer Exopeptidase. Im weiteren Verlauf wurden dann auch Kombinationen der einzelnen Carboxy- oder Aminopeptidasen untereinander betrachtet. Ziel sollte es sein durch sequenzielle oder parallele Applikation von Exopeptidasen unterschiedlicher Spezifität die zu erhaltende Sequenzinformation zu steigern.

Als letzter Aspekt seitens der Sequenzierung sollte noch die Anwendbarkeit auf Peptide mit post-translational-modifizierten Aminosäuren betrachtet werden. Eingehendere Versuche wurde dabei mit einigen Phosphopeptiden durchgeführt, da Phosphorylierungen eine sehr wichtige Rolle in der Regulation des Stoffwechselgeschehens spielen. Die schnelle Lokalisierung und Identifizierung solcher Modifikationen in Peptiden konnte bisher noch nicht zufriedenstellend gelöst werden. Parallel zum eigentlichen, enzymatischen Sequenzierungsvorgang sollten im Rahmen dieser Arbeit auch die Parameter der anschließenden MALDI-TOF Analyse untersucht und optimiert werden. Außerdem wurden zahlreiche Derivatisierungen im Hinblick auf eine Optimierung einzelner Teilaspekte der Leitersequenzierung untersucht. Mögliches Ziel solcher Derivatisierungen sollte die Verbesserung der gesamten, erarbeiteten Methodik sein, d.h. vom intakten Protein bis hin zur massenspektrometrischen Detektion der Leiterpeptide.

Im zweiten Teil der Arbeit sollte eine weitgehende Automatisierung der zuvor ausgearbeiteten Sequenzierungsmethode realisiert werden. Zu diesem Zweck sollten auf einem kommerziellen Pipettierroboter ein Testprotokoll ausgearbeitet werden, das alle Teilschritte einer komplexen Methode zur enzymatischen Sequenzierung integriert. Auch die bei der Probenbearbeitung anfallenden Schritte der Dateneingabe und die anschließende Datenweitergabe von Probenroboter an das Massenspektrometer sollten automatisch ablaufen. Abschließen wurde im Rahmen dieser Arbeit noch die Möglichkeit einer Automatisierung der massenspektrometrischen Messung der sequenzierten Proben untersucht.

Kapitel II:

Enzymatische Leitersequenzierungen

1 Theoretische Grundlagen

1.1 Grundlagen der Enzymatischen Sequenzierung

1.1.1 Exopeptidasen

Exopeptidasen fallen bezüglich ihrer Einteilung im Nomenklaturschema der IUPAC in die Unterklasse EC 3.4. Diese Unterklasse umfasst Hydrolasen, welche die Spaltung von Peptidbindungen katalysieren. Die unterschiedliche Spezifität der bekannten Exopeptidasen eröffnet theoretisch eine Fülle von Sequenzierungsmöglichkeiten. Eine Einteilung der Exopeptidasen erfolgt aus Sicht der Leitersequenzierung zweckmäßig nach der Angriffsseite des Enzyms am Peptid sowie der Zahl der in einem Abbauschritt abgespaltenen Aminosäuren (Tabelle 1).

Tabelle 1: Einteilung der Exopeptidasen für die Leitersequenzierung¹

C-terminale Sequenzierung	N-terminale Sequenzierung
<p><i>Monopeptidyl-Carboxypeptidasen</i></p> $H_2N - R_n - Xcc - Xbb \xrightarrow{\downarrow} Xaa - COOH$ <p>EC 3.4.16 Carboxypeptidasen von Serintyp EC 3.4.17 Metallo-carboxypeptidasen EC 3.4.18 Carboxypeptidasen vom Cysteintyp</p>	<p><i>Monopeptidyl-Aminopeptidasen</i></p> $H_2N - Xaa \xrightarrow{\downarrow} Xbb - Xcc - R_n - COOH$ <p>EC 3.4.11 Aminopeptidasen</p>
<p><i>Dipeptidyl-Carboxypeptidasen</i></p> $H_2N - R_n - Xcc \xrightarrow{\downarrow} Xbb - Xaa - COOH$ <p>EC 3.4.15 Peptidyl-Dipeptidasen</p>	<p><i>Dipeptidyl-Aminopeptidasen</i></p> $H_2N - Xaa - Xbb \xrightarrow{\downarrow} Xcc - R_n - COOH$ <p>EC 3.4.14 Dipeptidyl-Peptidasen</p>
	<p><i>Tripeptidyl-Aminopeptidasen</i></p> $H_2N - Xaa - Xbb - Xcc \xrightarrow{\downarrow} R_n - COOH$ <p>EC 3.4.14 Tripeptidyl-Peptidasen</p>
<p><i>Xaa, Xbb, Xcc: Aminosäuren</i></p> <p><i>R_n: restliche Peptidkette</i></p>	

¹ EC-Nomenklatur nach Empfehlungen der NC-IUBMB und IUPAC-IUBMB, Stand: September 2000
[Moss_1 2000]

Des weiteren sind die Omega-Peptidasen (EC 3.4.19) zu nennen, welche meist nur speziell modifizierte oder verknüpfte Aminosäuren von den Termini freisetzen, z.B. N-acetylierte Aminosäuren, Pyroglutamat oder γ -verknüpfte Glutaminsäure. Auch Dipeptidasen (EC 3.4.13), welche nur die Spaltung von Dipeptiden katalysieren, stellen einen Spezialfall dar. Aus Sicht der Leitersequenzierung mittels MALDI-TOF fallen Dipeptide in einen analytisch nicht nutzbaren Bereich für die Detektion (siehe Punkt 1.2.2.3 „Analytisch nutzbarer Bereich des Massenspektrums“). Daher ist diese Enzymklasse für diese Methode allgemein nicht von Bedeutung. Anders verhält es sich mit den Omega-Peptidasen, zu denen beispielsweise die Acylaminoacyl-Peptidase (EC 3.4.19.1) oder die Pyroglutamat-Peptidase I (EC 3.4.19.3) gehören. Insbesondere N-acetylierte Termini kommen in Proteinen als post-translationale Modifikation sehr häufig vor. In einigen Fällen der zu sequenzierenden Peptide einer proteolytischen Spaltung ist daher die Anwendung von Omega-Proteasen als erster Schritt durchaus sinnvoll. Für die Entwicklung der allgemeinen Methode sollten solche Spezialfälle jedoch zunächst nicht behandelt werden. Außerdem sollte bei einer Multi-Enzym-Methode, d.h. einer Methode die mit einer Kombination von Peptidasen arbeitet, ein zusätzlich vor- oder zwischengeschalteter Abbauschritt mit einer Omega-Peptidase kein Problem darstellen. Alle nachfolgenden Entwicklungen und Optimierungen der Methode sollten zunächst auf die 20 natürlichen, nicht-modifizierten Aminosäuren beschränkt bleiben. Anschließend musste dann untersucht werden, ob eine Erweiterung der Methode im Hinblick auf die Sequenzierung post-translational modifizierter Aminosäuren möglich ist und wie sich diese potentielle Erweiterung in die entwickelte Methode integrieren lässt.

Die Sequenzierungen sollten in dieser Arbeit ausschließlich mit Mono-peptidyl-Peptidasen (C-terminal) bzw. Peptidyl-Mono-peptidasen (N-terminal) erfolgen. Ziel dieser Strategie war eine möglichst einfache Interpretation der Ergebnisse der Aminosäureabspaltung mit direktem Rückschluss auf die Sequenzabfolge. Mit Peptidasen, die zwei oder mehrere Aminosäuren gleichzeitig abspalten, ist eine Sequenzierung zwar prinzipiell auch möglich, die Interpretation der Abspaltung und der Rückschluss auf die Aminosäuresequenz sind jedoch bei weitem nicht so einfach und eindeutig [Callahan 1972, Seidl 1989].

Aus dem Bereich der einfach abspaltenden Carboxy- und Aminopeptidasen ist jedoch nur ein Teil für eine Sequenzanalyse geeignet. Das entscheidende Kriterium ist dabei die Spezifität der Peptidase. Da bei einem Peptid unbekannter Sequenz mit jeder nur denkbaren Kombination der Aminosäuren gerechnet werden muss, sollte eine geeignete Peptidase ein möglichst breites Abspaltungsspektrum besitzen, d.h. möglichst unspezifisch sein. Peptidasen

die nur einen eingeschränkten Substratbereich umsetzen, wie z.B. Dipeptidasen sind daher ungeeignet. Eine ideale Peptidase sollte möglichst folgende Eigenschaften aufweisen:

- ❖ Abspaltung aller 20 natürlichen Aminosäuren
- ❖ Abspaltungskinetik für alle abbaubaren Aminosäuren vergleichbar

Tabelle 2 und Tabelle 3 geben einen Überblick über geeignete Exopeptidasen und deren Spezifitäten aus dem Bereich möglichen Peptidasen EC 3.4.11 bis EC 3.4.19 (siehe oben). Die beiden oben genannten, idealen Anforderungen werden von keiner der bisher bekannten Monocarboxy- oder Monoaminopeptidasen (gleichzeitig) erfüllt.

Tabelle 2: Geeignete Mono-peptidyl-Carboxypeptidasen ^{1, 2, 3}

Empfohlener Name (Fettdruck) Alternativer Name (u.a.)	EC	Familie	Spezifität	pH- Optimum	Temp.- Optimum
Gruppe I a: sehr unspezifische Carboxypeptidasen					
Carboxypeptidase C Carboxypeptidase Y Serin Carboxypeptidase I	3.4.16.5	S10	Xaa: relativ unspezifisch Xbb: unspezifisch aber Einfluss auf die Hydrolyserate	5,5 – 7	40 – 50°C
Carboxypeptidase D Cereal Serin Carboxypeptidase II	3.4.16.6	S10	Xaa: unspezifisch, bevorzugt Lys, Arg	5,5 – 6	keine Angabe
Carboxypeptidase Taq	3.4.17.19	M32	Xaa: unspezifisch, aber ≠ Pro Xbb: unspezifisch	8	80
Cathepsin X Cystein-Typ Carboxypeptidase	3.4.18.1	-	Xaa = unspezifisch, aber ≠ Pro Xbb: ≠ Pro	4 – 5,5	keine Angabe
Gruppe II a: relativ unspezifische Carboxypeptidasen					
Carboxypeptidase A	3.4.17.1	M14	Xaa: relativ unspezifisch, bevorzugt hydrophobe und neutrale AS keine/langsame Abspaltung von: Xaa = Asp, Glu, Arg, Lys, Pro Xbb: unspezifisch	7 – 9	25-37°C (Assay)
Carboxypeptidase A2	3.4.17.15	M14	Xaa: wie Carboxypeptidase A relativ unspezifisch mit Bevorzugung voluminöserer Reste – aromatisch oder aliphatisch Xbb: unspezifisch	7,5 (Assay)	25°C (Assay)
Carboxypeptidase T	3.4.17.18	M14	Xaa: relativ unspezifisch hydrophob oder positiv geladen Xbb: unspezifisch	7 – 8	keine Angabe

Empfohlener Name (Fettdruck) Alternativer Name (u.a.)	EC	Familie	Spezifität	pH- Optimum	Temp.- Optimum
Gruppe III a: relativ spezifische Carboxypeptidasen					
Carboxypeptidase H Carboxypeptidase E	3.4.17.10	M14	Xaa: Lys, Arg (His) Xbb: unspezifisch	5 – 6	37°C (Assay)
Carboxypeptidase B Protaminase	3.4.17.2	M14	Xaa: Lys, Arg teilweise auch hydrophobe AS Xbb: unspezifisch	7 – 9	25-37°C (Assay)
Carboxypeptidase U Carboxypeptidase R	3.4.17.20	M14	Xaa: Arg, Lys Xbb: unspezifisch	7,5 – 8	keine Angabe
Metallo-carboxypeptidase D	3.4.17.22	M14	Xaa: Arg, Lys Xbb: unspezifisch	5,5 - 6,5	keine Angabe
Carboxypeptidase M	3.4.17.12	M14	Xaa: Lys, Arg Xbb: unspezifisch	6,5 – 8	37°C (Assay)
Lysin Carboxypeptidase Carboxypeptidase N	3.4.17.3	M14	Xaa: Lys (Arg) Xbb: ≠ Pro, sonst unspezifisch aber Einfluss auf die Hydrolyserate	7 – 7,5	37°C (Assay)

Gruppe IV a: spezifische Carboxypeptidasen					
Lysosomal Pro-X Carboxypeptidase Carboxypeptidase P Angiotensinase C	3.4.16.2	S28	Xaa: unspezifisch Xbb: Pro	4,5-5,5	keine Angabe
Gly-X Carboxypeptidase Glycin Carboxypeptidase Carboxypeptidase S	3.4.17.4	M20	Xaa: unspezifisch Xbb: Gly	6	37°C (Assay)
Alanin Carboxypeptidase N-Benzoyl-L-Alanin-Aminohydrolase	3.4.17.5	-	Xaa: Ala Xbb: unspezifisch	7 – 8	30°C (Assay)
Membran Pro-X Carboxypeptidase Carboxypeptidase P	3.4.17.16	-	Xaa: unspezifisch, aber ≠ Pro Xbb: bevorzugt Prolin	7,8	keine Angabe
Glutamat Carboxypeptidase Gamma-glutamyl Hydrolase Carboxypeptidase G	3.4.17.11	M20	Xaa: unspezifisch Xbb: Glu (Gamma-glutamyl Bindung zu Xaa)	6 – 8	30 – 42°C

¹ Daten zusammengetragen aus NC-IUBMB und BRENDA, Stand: September 2000 [Moss 2000, Schomburg 2000]

² Symbole und Abkürzungen der Aminosäuren: siehe Anhang D

³ Zur Erklärung von Xaa, Xbb, Xcc siehe Tabelle 1 (S. 14)

Tabelle 3: Geeignete Mono-peptidyl-Amino-peptidasen ^{1, 2, 3}

Empfohlener Name (Fettdruck) Alternativer Name (u.a.)	EC	Familie	Spezifität	pH- Optimum	Temp.- Optimum
Gruppe I b :sehr unspezifische Amino-peptidasen					
Membrane Alanyl Amino-peptidase Microsomal Amino-peptidase Amino-peptidase M	3.4.11.2	M1	Xaa: unspezifisch, auch Pro bevorzugt Ala Xbb: unspezifisch	7 – 8	40 – 60°C
Clostridial Amino-peptidase <i>Clostridium Histolyticum</i> Amino-peptidase	3.4.11.13	-	Xaa: unspezifisch, auch Pro, Hyp Xbb: unspezifisch, aber ≠ Pro	8	keine Angabe

Empfohlener Name (Fettdruck) Alternativer Name (u.a.)	EC	Familie	Spezifität	pH- Optimum	Temp.- Optimum
Cytosol Alanyl Aminopeptidase Cytosol Aminopeptidase III	3.4.11.14	M1	Xaa: unspezifisch, bevorzugt Ala Xbb: unspezifisch, aber ≠ Pro	8,5	keine Angabe
Aminopeptidase Ey	3.4.11.20	M1	Xaa: unspezifisch Xbb: unspezifisch	6,5 – 8	37°C (Assay)
Aminopeptidase I Yeast Aminopeptidase I	3.4.11.22	M18	Xaa: unspezifisch, bevorzugt neutrale oder hydrophobe AS Xbb: unspezifisch	6 – 8,5	keine Angabe
Gruppe II b: relativ unspezifische Aminopeptidasen					
Leucyl Aminopeptidase Cytosol Aminopeptidase	3.4.11.1	M17	Xaa: unspezifisch, auch Pro bevorzugt Leu, aber ≠ Arg, Lys Xbb: unspezifisch, aber ≠ Pro hat Einfluss auf die Hydrolyserate	7 – 10	26 – 40°C
Bacterial Leucyl Aminopeptidase <i>Aeromonas Proteolytica</i> Aminopeptidase	3.4.11.10	M17 oder M28	Xaa: unspezifisch, bevorzugt Leu aber ≠ Glu, Asp Xbb: unspezifisch, aber ≠ Pro	8	40°C
Gruppe III b: relativ spezifische Aminopeptidasen					
Aminopeptidase B Arginin Aminopeptidase	3.4.11.6	M1	Xaa: Lys, Arg Xbb: relativ unspezifisch	6,5 – 7,5	28 – 37°C
Glutamyl Aminopeptidase Aminopeptidase A	3.4.11.7	M1	Xaa: Glu, Asp Xbb: unspezifisch	6,5 – 8	50 – 55°C
Aminopeptidase Y Lysyl Aminopeptidase	3.4.11.15	M28	Xaa: Lys, Arg Xbb: unspezifisch	8,5	keine Angabe
Aspartyl Aminopeptidase	3.4.11.21	M18	Xaa: Asp, Glu Xbb: unspezifisch	keine Angabe	keine Angabe
Gruppe IV b: spezifische Aminopeptidasen					
Prolyl Aminopeptidase Prolin Iminopeptidase	3.4.11.5	S33	Xaa: Pro Xbb: unspezifisch aber Einfluss auf die Hydrolyserate	7 – 8	50°C
X-Pro Aminopeptidase Prolin Aminopeptidase Aminopeptidase P	3.4.11.9	M24	Xaa: unspezifisch, auch Pro hat Einfluss auf die Hydrolyserate Xbb: Pro	7,5 – 9	keine Angabe
X-Trp Aminopeptidase Aminopeptidase W	3.4.11.16	-	Xaa: unspezifisch, bevorzugt Glu, Leu Xbb: Trp, weniger gut Phe, Tyr	7,5	keine Angabe
Tryptophanyl Aminopeptidase	3.4.11.17	-	Xaa: Trp Xbb: unspezifisch	9,0 – 9,5	40 – 45°C
Methionyl Aminopeptidase Peptidase M	3.4.11.18	M24	Xaa: Met Xbb: Gly, Ala, Val, Ser, Thr Xbb ≠ Arg, Lys, Asp, Asn, Glu, Gln, Leu, Ile, Met	6 – 8	30 – 45°C

¹ Daten zusammengetragen aus NC-IUBMB und BRENDA, Stand: September 2000 [Moss 2000, Schomburg 2000]

² Symbole und Abkürzungen der Aminosäuren: siehe Anhang D

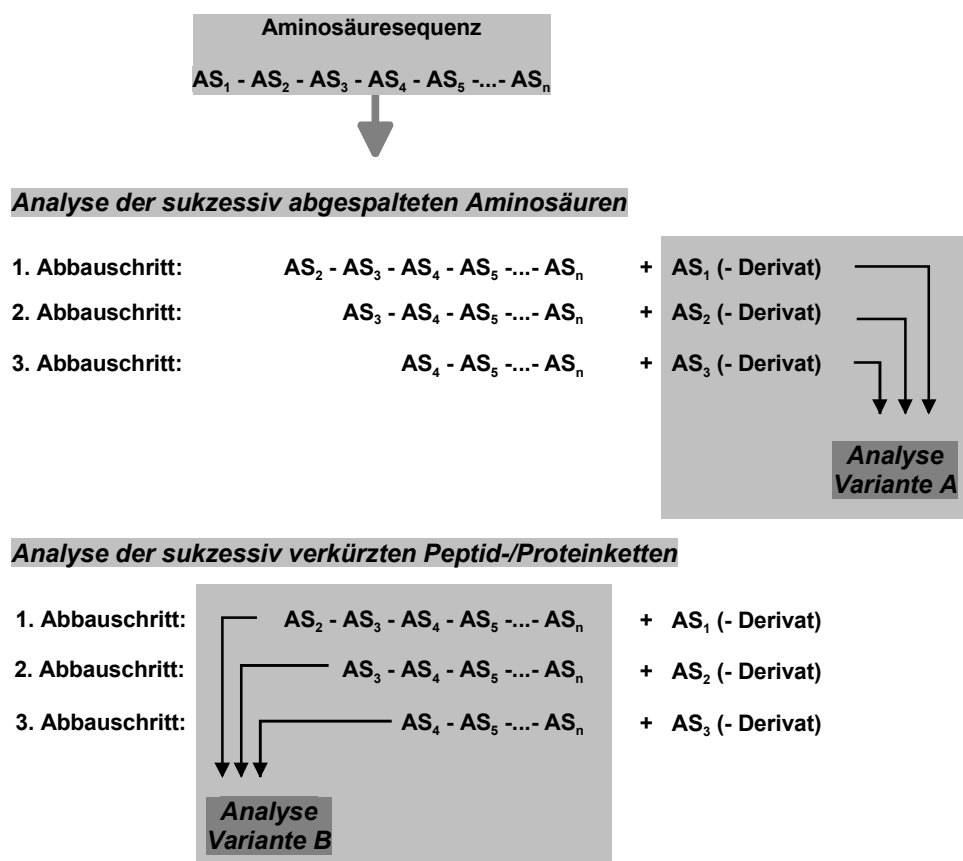
³ Zur Erklärung von Xaa, Xbb, Xcc siehe Tabelle 1 (S. 14)

Die Einteilung der Peptidasen in Tabelle 2 und Tabelle 3 erfolgt zwar willkürlich, jedoch aus der Sicht der Sequenzierung zweckmäßig nach dem Grad der Spezifität. Die beiden Tabellen zeigen, dass sich bereits mit den derzeit bekannten Exopeptidasen theoretisch nahezu alle möglichen Peptide vollständig sequenzieren lassen, sofern sie aus den 20 natürlichen Aminosäuren aufgebaut sind. Allerdings sind nur die wenigsten der hier angegebenen Peptidasen auch kommerziell erhältlich. Nur die in der Tabelle grau unterlegten Peptidasen sind von einem oder mehreren Anbietern käuflich zu erwerben und wurden auch im Rahmen dieser Arbeit verwendet oder zumindest getestet. Hier ist jedoch in einigen Fällen, wie z.B. bei EC 3.4.16.5 (Carboxypeptidase C), das Enzym aus mehreren verschiedenen Spezies oder Geweben erhältlich. Wie sich im Laufe der Arbeit herausstellte, können hier teilweise starke Unterschiede im Abbauverhalten festgestellt und genutzt werden.

1.1.2 Möglichkeiten zur Verfolgung der Aminosäureabspaltung

Für die Beobachtungsform der Aminosäureabspaltung, die einen Rückschluss auf die Primärsequenz zulässt, sind zwei Varianten denkbar. Diese basieren auf den unterschiedlichen Produkten der Sequenzierungsreaktion und sind in Abbildung 2 zusammengefasst.

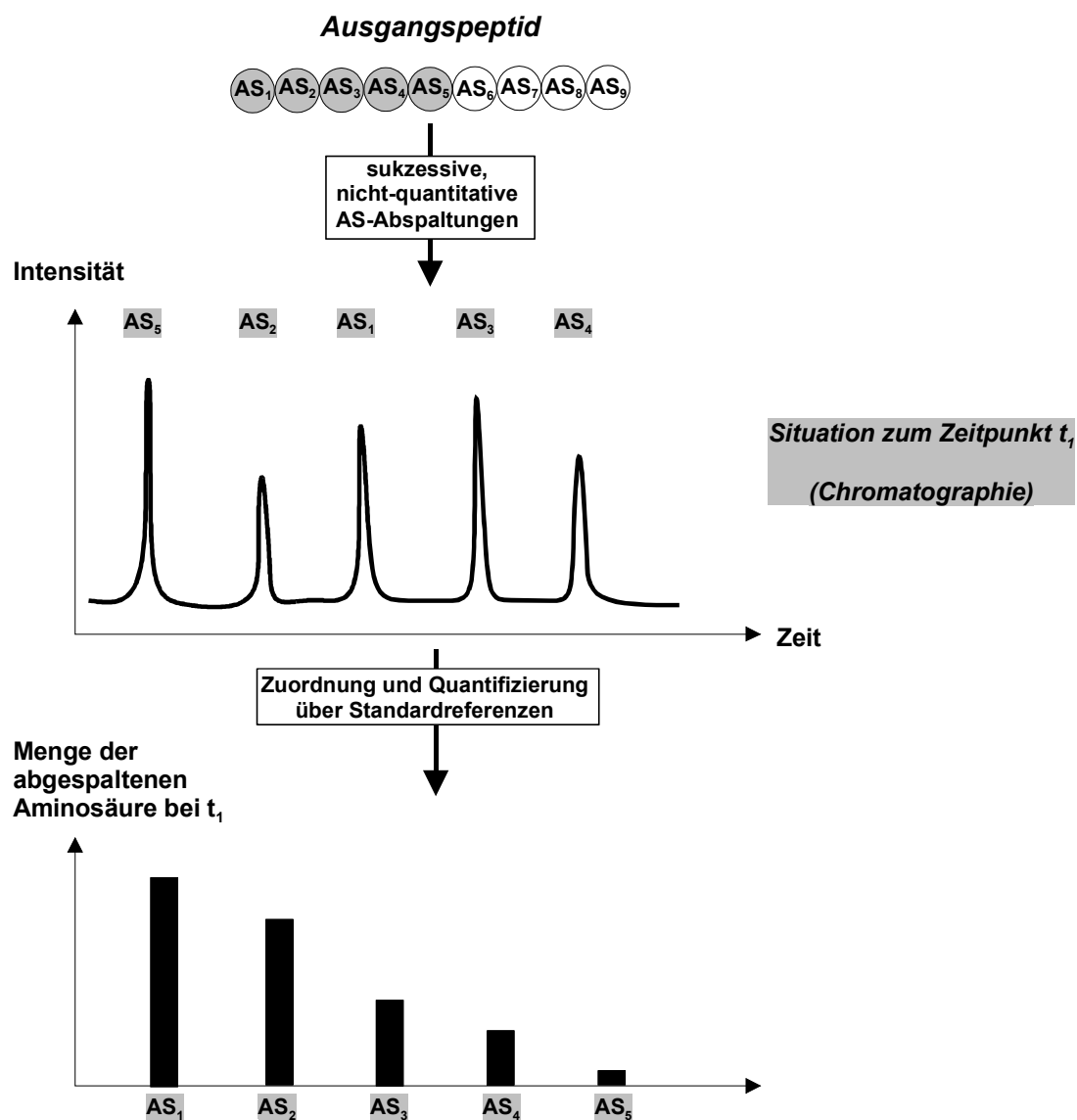
Abbildung 2: Möglichkeiten zur Verfolgung der Aminosäureabspaltung



AS_x = Aminosäure an Position X

Variante A entspricht dem Vorgehen bei den chemischen Sequenzierungen nach Edman oder Schlack-Kumpf. Auch hier werden die sukzessiv vom Peptid- oder Protein abgespaltenen Aminosäuren untersucht. Die enzymatische Sequenzierung arbeitet jedoch nicht, wie eine der beiden vorgenannten Methoden, in Cyclen. Die Sequenzinformation wird erhalten, indem die abgespaltenen Aminosäuren in ihrer Menge zu einem oder mehreren Zeitpunkten bestimmt werden. Diese Form der enzymatischen Sequenzierungstechnik war die Erste, die für C- und N-terminale enzymatische Sequenzierungen genutzt wurde, z.B. von [Tschesche 1977] oder [Light 1972]. Abbildung 3 zeigt schematisch ein mögliches Analysenergebnis, wenn zu einem bestimmten Zeitpunkt t_1 eine Probe aus dem Spaltansatz entnommen, abgestoppt und analysiert wird.

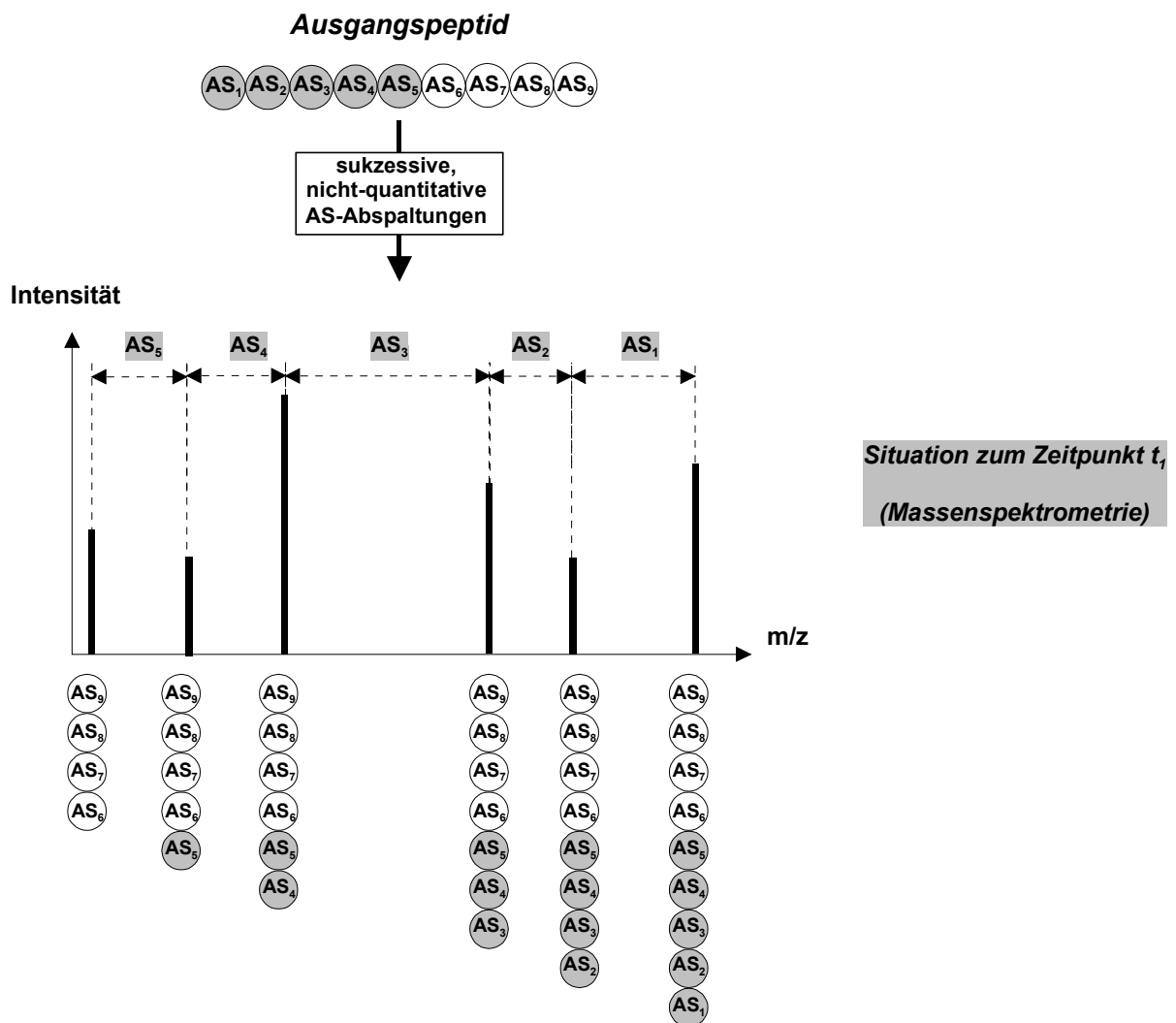
Abbildung 3: Verfolgung der Aminosäureabspaltung über die abgespaltenen Aminosäuren



Für die Analytik der abgespaltenen Aminosäuren sind dabei mehrere Varianten möglich. Die bekannteste Methode stellt hierbei die Trennung der nicht-modifizierten Aminosäuren über Ionenaustausch, mit anschließender Nachsäulenderivatisierung (z.B. mit Ninhydrin, Fluorescamin oder ortho-Phthaldialdehyd) nach Stein und Moore dar [Moore 1948, Manning 1993]. Alternativ dazu besteht die Möglichkeit einer Vorsäulenderivatisierung der abgespaltenen Aminosäuren (z.B. mit ortho-Phthaldialdehyd [Alvarez 1989], Phenylisothiocyanat [Heinrikson 1984], Fluorenylmethoxycarbonylchlorid [Einarsson 1983], Dansylchlorid [Tapuhi 1981] oder Dabsylchlorid [Lin 1975]). Die Auftrennung der Derivate erfolgt dann über RP-HPLC. Zur Identifizierung einer bestimmten Aminosäure dient die Retentionszeit im Vergleich zu einer Referenzverbindung. Auch eine Elektrophorese auf Filterpapier mit verschiedenen Färbungsmethoden eignet sich zu diesem Zweck für die Aminosäureanalytik [Ambler 1972]. Die Information über die Sequenzabfolge wird bei allen genannten Methoden durch die quantitativen Mengenverhältnisse der einzelnen Aminosäuren zueinander festgelegt. Diese Quantifizierung erfolgt über Absorptions- oder Densitometrische Messung und Vergleich mit Standardreferenzsubstanzen. Insbesondere mit der Fluorometrie kann hier sehr empfindlich gearbeitet werden. Was jedoch eindeutig gegen Variante A zur Sequenzbestimmung spricht, ist der hohe Arbeits- und Zeitaufwand durch die notwendige Aminosäuretrennung, eventuelle Derivatisierung und Quantifizierung über Standards.

Variante B entspricht der Leitersequenzierung und stellt mit den heute zur Verfügung stehenden Analysemethoden die wesentlich elegantere Variante dar. Abbildung 4 (nächste Seite) zeigt das Analysenprinzip von Variante B. Die Detektion der sukzessiv verkürzten Peptidfragmente erfolgt am einfachsten zu einem (t_I) oder mehreren Zeitpunkten massenspektrometrisch. Der Vorteile gegenüber Variante A liegt in der Tatsache, dass zur Rekonstruktion der Sequenz weder eine Trennung der Reaktionsprodukte (Aminosäuren und verkürzte Peptidketten), noch eine quantitative Erfassung der abgespaltenen Aminosäuren notwendig ist. Die Teilsequenz kann direkt aus der Mischung ausgelesen werden. Die Identifizierung der abgespaltenen Aminosäuren erfolgt über die Massendifferenzen der konsekutiven Restpeptidmassen im Spektrum, die den Massen der abgespaltenen Aminosäuren entsprechen. Voraussetzung für eine erfolgreiche Interpretation des Spektrums ist, dass alle Massendifferenzen einer Aminosäuren eindeutig zugeordnet werden können.

Abbildung 4: Verfolgung der Aminosäureabspaltung indirekt über die sukzessive verkürzten Peptidfragmente



Alternative Detektionsarten sind bei beiden Variante denkbar, also z.B. Massenspektrometrie der Aminosäuren für Variante A oder HPLC der Leiterpeptide für Variante B, bringen jedoch entweder keinen Vorteil oder sind nur schwer realisierbar. Eine massenspektrometrische Detektion bei Variante A beispielsweise erfordert auch weiterhin eine Quantifizierung der Aminosäuren und leidet unter dem Problem einer einfachen massenspektrometrischen Technik für die Analyse der Aminosäuren.

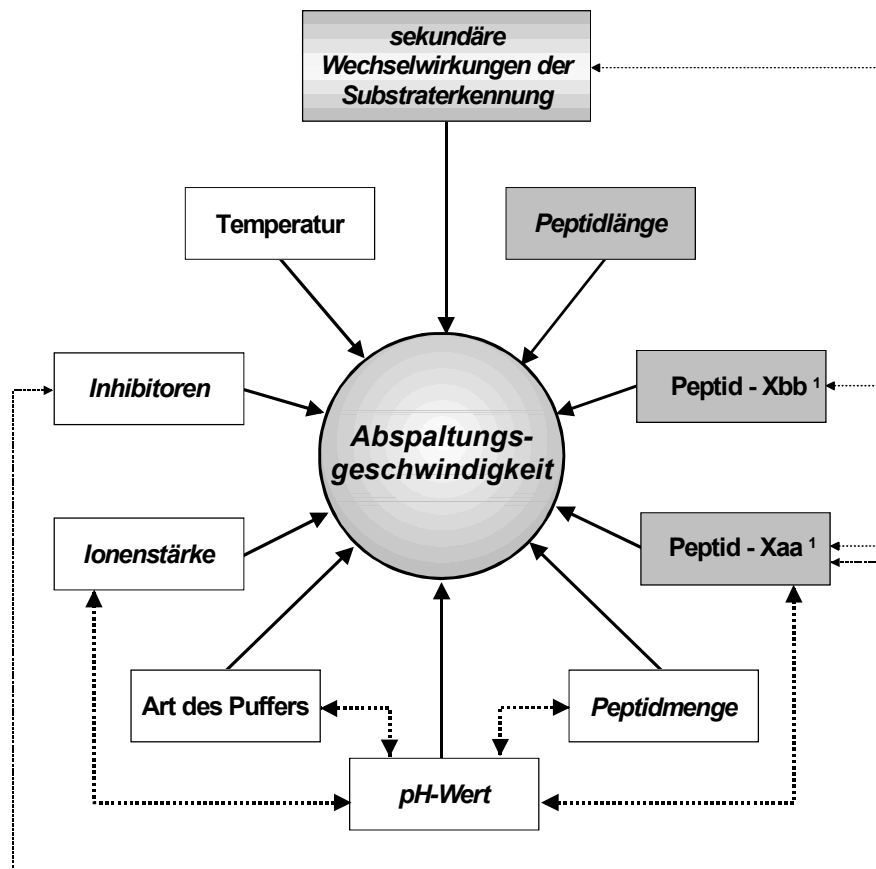
1.1.3 Kinetische Steuerung der Aminosäureabspaltung

Die unter 1.1.1 für die einzelnen Peptidasen angegebenen Spaltungsspezifitäten sind meist nicht eindeutig abzugrenzen. Die Abspaltungsgeschwindigkeiten einzelner Aminosäuren bei unspezifischen Peptidasen unterscheiden sich oft erheblich voneinander. Abbildung 5 zeigt

eine Auswahl wichtiger Faktoren, welche die Abspaltungskinetik der Aminosäuren bei der Sequenzierung mit Exopeptidasen beeinflussen können.

Einige der Einflussparameter besitzen auch Abhängigkeiten untereinander, so dass ein kompliziertes Netzwerk entsteht, welches die Optimierung einer kinetischen Steuerung bei der Sequenzierung stark erschwert. Beispielsweise wirken vielen Fällen die abgespaltenen Aminosäuren als Inhibitoren für die abspaltende Exopeptidase. Zum Beispiel inhibiert Phenylalanin, das durch CPA abgespalten wird diese Exopeptidase [Schomburg 2000]. In den Organismen, in denen diese Exopeptidasen vorkommen, ist eine solche kompetitive Hemmung durch ein Reaktionsprodukt häufig ein wichtiger regulatorischer Mechanismus.

Abbildung 5: mögliche Einflussfaktoren auf die Abspaltungsgeschwindigkeit von Aminosäuren



¹ Bezeichnungen und Bedeutung von Xaa, Xbb: siehe Tabelle 1 (S. 14)

Für eine Optimierung der Sequenzierung muss zwischen zwei Arten von Einflussparametern auf die Abbaukinetik unterschieden werden:

- ❖ Parameter, die eine aktive Kontrolle der Abspaltungsgeschwindigkeit erlauben
- ❖ Parameter, die sich nicht durch Steuerung von außen beeinflussen lassen

Zur letzteren Kategorie zählen die in Abbildung 5 grau-massiv unterlegten Felder. Insbesondere die abzusplattende Aminosäure Xaa am Peptid hat auf die Abbaugeschwindigkeit meist einen starken Einfluss. Bei Realproben, in denen neben den im Peptid vorhandenen Aminosäuren auch die Peptidmenge unbekannt ist, gestaltet sich die Kontrolle des Abbaus somit schwierig. Die Rolle von Xaa und Xbb bei der Abspaltung einer Aminosäure fällt unter den Begriff der sekundären Wechselwirkungen der Substraterkennung [Schechter 1970, Umetsu 1997]. In diese Wechselwirkungen kann ein weit größerer Sequenzabschnitt des Peptidsubstrats involviert sein, so dass neben Xaa und Xbb noch weitere Aminosäuren des Peptids Einfluss auf die Abspaltungskinetik besitzen, z.B. über die Substratbindung.

Parameter der ersten Kategorie (weiße Felder in Abbildung 5), die üblicherweise zur Kontrolle der Peptidaseaktivität eingesetzt werden, sind in erster Linie pH und Temperatur. Gerade der pH-Wert ist jedoch mit anderen Parametern stark vernetzt. Vor allem ist der optimale pH bei den meisten Exopeptidasen auch abhängig von der abzusplattenden Aminosäure Xaa – also einer unbekannten Größe. Die nachfolgende Aufstellung zeigt einige der hier diskutierten Abhängigkeiten der Abspaltungsgeschwindigkeit am Beispiel der Carboxypeptidase Y (EC 3.4.16.), welche auch in dieser Arbeit verwendet wurde.

Abbaugeschwindigkeit für Xaa mit CPY:

schnell:	Phe, Tyr, Trp, Leu, Ile, Val, His
gut:	Ser, Thr, Met, Ala, Asn, Glu, Gln, Lys, Arg, Pro
langsam:	Gly, Asp

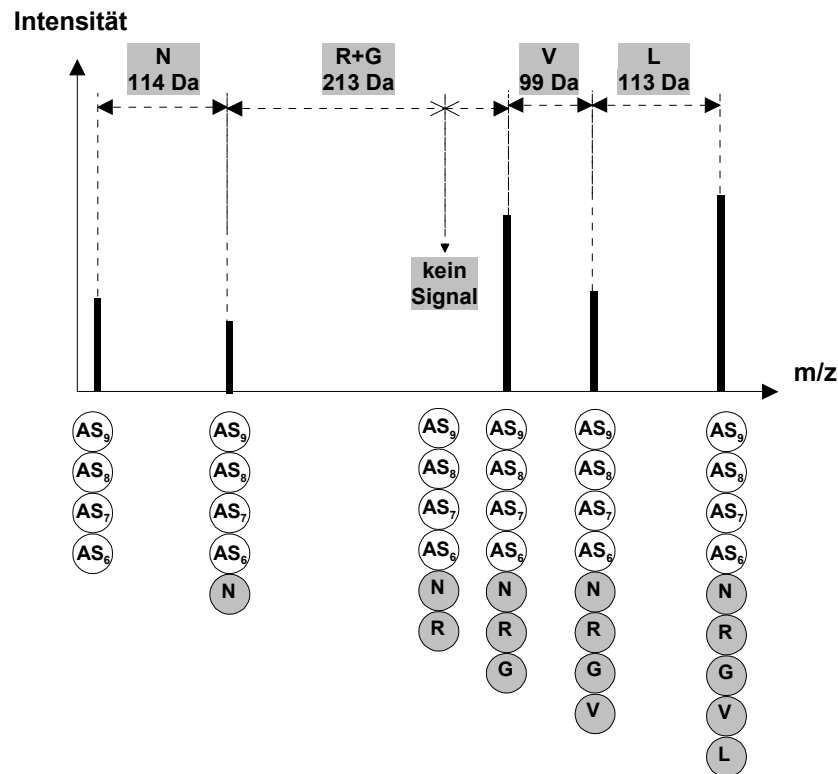
pH-Optima (Bereiche) für den Abbau bestimmter Xaa mit CPY:

pH 5,5 und kleiner:	saure Aminosäuren
pH 6:	neutrale Aminosäuren
pH 6 – 7:	basische Aminosäuren

Als eine Folge der ausgeprägten, kinetischen Unterschiede können bestimmte Leiterpeptide während des Abbauprozesses in nur sehr geringen Mengen vorhanden sein. Dieser Fall tritt dann ein, wenn z.B. bei N-terminalem Abbau der Sequenz $H_2N\text{-}Xaa\text{-}Xbb\text{-}R_n\text{-}COOH$ die Aminosäure Xaa nur langsam, die nachfolgende Aminosäure Xbb dagegen relativ schnell abgebaut wird. Wenn das Zwischenprodukt $H_2N\text{-}Xbb\text{-}R_n\text{-}COOH$ nach der Abspaltung von Xaa nur in sehr geringer Menge vorliegt, kann das entsprechende Signal im Leiterspektrum

fehlen. In diesem Fall bezeichnet man die Stelle mit dem fehlenden Leiterpeptidsignal im Leiterspektrum als, sogenanntes „Gap“ („Lücke“). Im Falle der C-terminalen Sequenzierung gilt dies analog für die Sequenz $H_2N-R_n-Xbb-Xaa-COOH$). Abbildung 6 zeigt ein Beispiel für Carboxypeptidase Y: In diesem Fall kann sowohl bezüglich der abgebauten Aminosäuren, als auch bezüglich deren Abfolge eine Mehrdeutigkeit auftreten. Die gefundene Massendifferenz von 213 Da über der Sequenzlücke ist für die Kombination [Gly/Arg] nicht spezifisch, da diese Summe zum Beispiel auch für das Paar [Val/Asn] zutrifft. Auch bezüglich der Sequenzabfolge sind alle Permutation denkbar, also neben Gly-Arg auch Arg-Gly und neben Val-Asn auch Asn-Val.

Abbildung 6: Interpretationsprobleme bei Sequenzlücken in Leiterspektrum



Im ungünstigsten Fall ist der Abbau an einer bestimmten Sequenzposition des Peptids so stark gehemmt (Abbau der betreffenden Aminosäure extrem langsam oder nicht möglich), dass die Sequenzierung hier praktisch abbricht. Dieser Fall kann eintreten, wenn beispielsweise bei N-terminaler Sequenzierung des Peptids $H_2N-Xaa-Xbb-R_n-COOH$ der Abbau der Aminosäure *Xaa* extrem stark gehemmt ist. Bei sehr geringen Konzentrationen der nachfolgenden Leiterpeptide endet die Leitersequenz dann mit dem Peptidsignal von $H_2N-Xaa-Xbb-R_n-COOH$.

Bei der Optimierung der Werte für pH und Temperatur einer Leitersequenzierung sind die Parameter so zu bestimmen, dass möglichst alle intermediären Leiterpeptide fassbar sind. Der Begriff „optimal“ bezeichnet hier also eine kontrollierte Kinetik und nicht eine möglichst schnelle Abspaltung aller Aminosäuren. Die diesbezüglichen pH-Werte aus Tabelle 2 und Tabelle 3 dienen daher für die Leitersequenzierung nur als Anhaltspunkte. Hinzu kommt die Tatsache, dass die meisten der in der Literatur bestimmten Werte nur für ein bestimmtes Substrat gelten, dass oft bezüglich seiner Größe und Struktur mit den für die Sequenzierung interessanten Peptiden von 10-30 Aminosäuren Länge nicht vergleichbar ist.

1.1.4 Peptidase-Kombinationen

Betrachtet man die Eigenschaften der Exopeptidasen in Tabelle 2 und Tabelle 3, so wird die geforderte, möglichst unspezifische Abspaltung aller Aminosäuren mit ähnlicher Geschwindigkeit bei den meisten zu sequenzierenden Peptiden mit einer einzigen Peptidase kaum realisierbar sein. Als Lösungsstrategie dieses Problems bietet sich die Kombination mehrerer Peptidasen an. Bezüglich der Anwendungsform einer solchen Peptidasenkombination stehen zwei Wege zur Wahl:

Kombinationsform A: sequenziell

Kombinationsform B: parallel

Bei einer sequenziellen Kombination werden Enzyme unterschiedlicher Spaltspezifität nacheinander zum Peptid gegeben. Der Vorteil ist, dass theoretisch für jede Peptidase auf die für die Sequenzierung günstigsten Bedingungen (insbesondere der pH-Wert) eingestellt werden kann. Die Bedingungen für eine möglicherweise notwendige Umpufferung vor der Zugabe nachfolgender Peptidasen können zuvor mit einem größeren Lösungsvolumen erarbeitet und anschließend auf die kleineren Volumina einer Mikromethode übertragen werden. Bei der sequenziellen Enzymkombination kann auch gezielt auf die durch die Proteinspaltung erzeugten Termini der proteolytisch oder chemisch erzeugten Fragmente eingegangen werden. Ein Beispiel bietet die Anwendung von Carboxypeptidase B (CPB) und Carboxypeptidase C (CPC) bei Peptiden, welche mit den Endoproteinasen LysC oder Trypsin erzeugt wurden. Hier wird bei der Leitersequenzierung zunächst das C-terminale Lysin oder Arginin (basische Aminosäuren) mit CPB abgespalten. Im zweiten Schritt können dann die Bedingungen der Sequenzierung mit CPC weitgehend auf hydrophobe, neutrale und saure Aminosäuren eingestellt werden.

Bei einem proteolytisch erzeugten Peptid aus einem unbekannten Protein kann jedoch nur für den durch die Spaltungsspezifität der Endopeptidase bestimmten Peptidterminus (z.B. C-terminales Arginin oder Lysin bei Spaltungen mit Trypsin) die Reihenfolge der sequenziellen Exopeptidaseauftragung derart systematisch angepasst werden. Die optimale Reihenfolge der Auftragung verschiedener Enzyme ist natürlich im allgemeinen zunächst unklar. Für die unbekannte Sequenz des proteolytisch erzeugten Peptids fehlt die Information, zu welchem Zeitpunkt der Abspaltung eine bestimmte Enzymspezifität von Vorteil oder gar gefordert ist. Eine Mischung der verschiedenen Enzyme, also deren parallele Auftragung (Kombinationsform B), ist daher prinzipiell sinnvoller. Hier steht zu jedem Zeitpunkt der Sequenzierung die für die Abspaltung notwendige Peptidase mit der erforderlichen Spezifität bereit. Problematisch bei der Anwendung von solchen „Enzym-Cocktails“ ist allerdings die Tatsache, dass ein Parametersatz wie in Abbildung 5 für alle in der Mischung enthaltenen Enzyme gemeinsam optimiert werden muss. Die Möglichkeit einen Kompromisses für gute Abspaltungsbedingungen einer solchen Peptidasemischung zu finden, sind jedoch durchaus gegeben, da geeignete Bedingungen für die Leitersequenzierung wie bereits erklärt nicht gleichzusetzen sind, mit den Bedingungen für maximale Abspaltungsgeschwindigkeit. Praktisch wird es jedoch schwierig sein, gleichermaßen günstige Abspaltungsbedingungen für Peptidasen zu finden, deren Optima z.B. im Hinblick auf Temperatur, pH-Wert oder kompatible Puffer stark voneinander abweichen. Gleichzeitig wird es auch mit zunehmender Zahl der kombinierten Enzyme schwieriger werden, die Bedingungen für alle Enzyme zu optimieren.

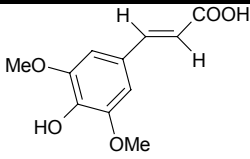
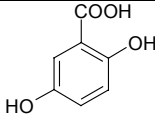
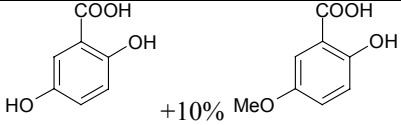
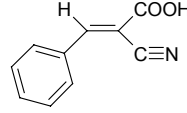
1.2 Grundlagen der Leiterpeptid-Massenanalyse mittels MALDI-TOF-MS

1.2.1 Prinzip der Massenanalyse mit MALDI-TOF

In den vergangenen Jahren hat sich die durch Karas und Hillenkamp [Karas 1988, Karas 1991] entwickelte Matrix-unterstützte Laserdesorptions/Ionisations-Massenspektrometrie (MALDI-MS) zu einer der wichtigsten Techniken der Molekülmassenbestimmung von Biomolekülen entwickelt [Gross 1998]. Eine massenspektrometrische Analyse der Leiterpeptide mittels MALDI-TOF erscheint vor allem hinsichtlich der einfachen Kopplung mit der enzymatischen Sequenzierung, aber auch bezüglich der hohen Sensitivität und Massengenauigkeit im Peptidbereich am geeignetsten. Zudem gestattet die fast ausschließliche Bildung einfach geladener, positiver oder negativer Ionen im MALDI-Prozess eine sehr einfache Spektreninterpretation der Leiterpeptidgemische.

Die Präparation der Probe erfolgt durch Mischen mit einem niedermolekularen Feststoff, der sogenannten Matrix. Bei den meisten verwendeten Matrices handelt es sich um organische Säuren. Tabelle 4 zeigt einige typische Beispiele für Matrixsubstanzen, die heute in der MALDI-MS von Proteinen und Peptiden routinemäßige Anwendung finden. Das molare Matrix-Analyt-Verhältnis sollte ca. $10^3:1$ bis $10^4:1$ betragen.

Tabelle 4: Beispiele gängiger Matrixsubstanzen, für die MALDI-MS von Proteinen und Peptiden

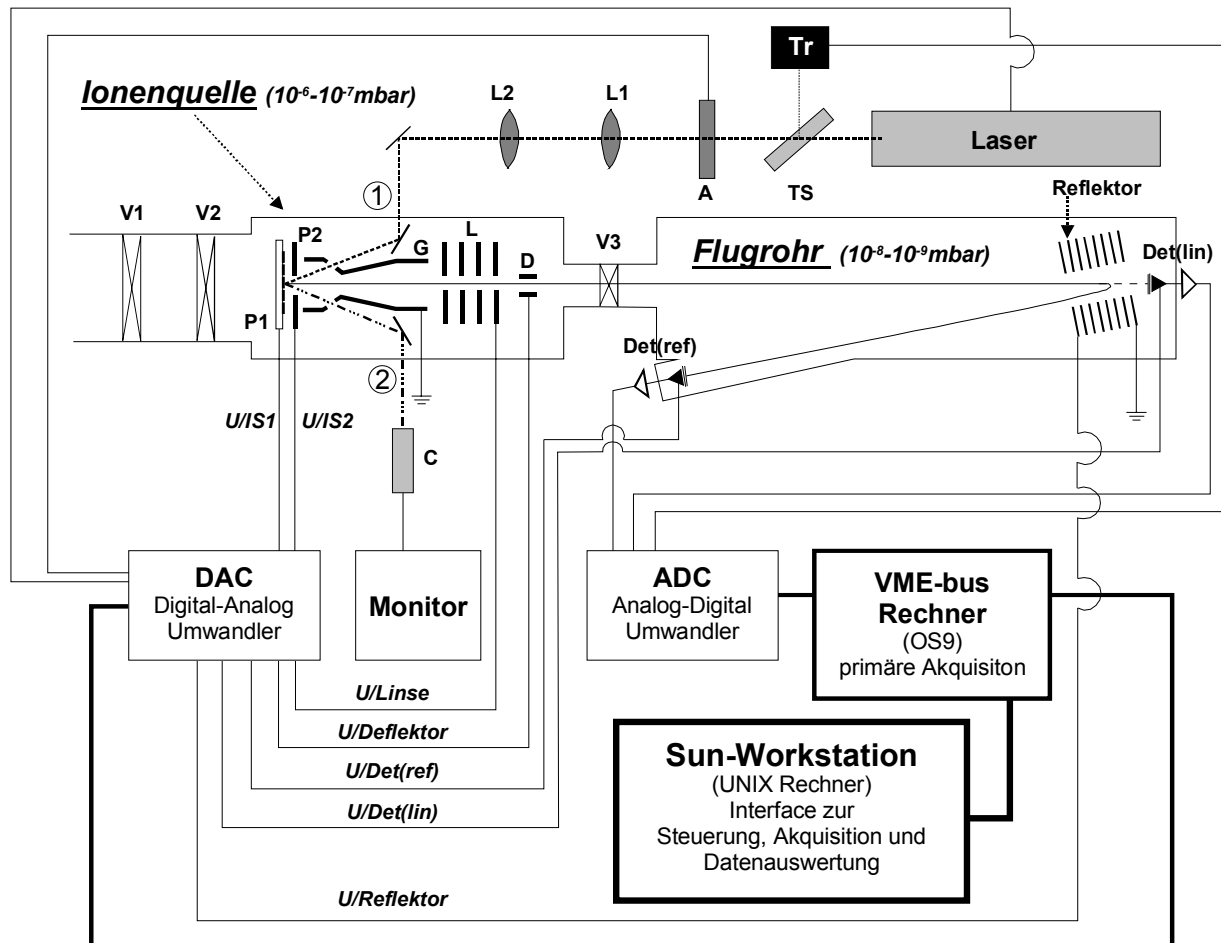
Name der Matrix (Abkürzung)	Strukturformel ¹	λ ²	Anwendungsbereich
3,5-Dimethoxy-4-hydroxyzimtsäure, Sinapinsäure (SA)		266 nm 337 nm 355 nm	Proteine
2,5-Dihydroxybenzoesäure (DHB)		266 nm 337 nm 355 nm	Proteine, Peptide
2,5-Dihydroxybenzoesäure +10% 5-Methoxysalicylsäure (DHBs)		266 nm 337 nm 355 nm	Proteine, Peptide
α -Cyano-4-hydroxycimtsäure (CHCA)		337 nm 355 nm	Peptide
Bernsteinsäure (BS)	HOOC-CH ₂ -CH ₂ -COOH	2,94 μ m 10,6 μ m	Proteine, Peptide

¹ OMe = Methoxy

² genutzte Wellenlänge(n) für die Laseranregung

Abbildung 7 zeigt den typischen Aufbau eines modernen MALDI-TOF-MS am Beispiel des Gerätes „Reflex III“ der Firma Bruker.

Abbildung 7: typischer Aufbau eines de-MALDI-re-TOF-Massenspektrometers (am Beispiel des „Reflex III“ der Firma Bruker)



- P1: Trägerplatte mit Probe ($U/IS1$ = Spannungsparameter)
P2: Elektrode für *delayed extraction* ($U/IS2$ = Spannungsparameter)
G: Gegenelektrode auf Erdpotential
L: Linse für Ionenfokussierung ($U/Linse$ = Spannungsparameter)
D: Ablenkelektrode (Deflektor) ($U/Deflektor$ = Spannungsparameter)
Det(ref): Detektor und Vorverstärker für den Reflektormodus ($U/Det(ref)$ = Spannungsparameter)
Det(lin): Detektor und Vorverstärker für den Linearmodus ($U/Det(lin)$ = Spannungsparameter)
Tr: Trigger
TS: Teildurchlässiger Spiegel
A: Laserstrahl-Abschwächer
L1, L2: Fokussierungslinsen für den Laserstrahl
V1, V2, V3: Vakuumschleusen
C: CCD-Kamera
① ----- Weg des Laserstrahls auf die Probe
② ----- Beobachtung der Probe mittels CCD-Kamera/Monitor

An der Stelle P1 befindet sich bei der Messung der Träger (MALDI-Target) mit der präparierten Probe im Vakuum der Ionenquelle. Der Matrix kommt zunächst für den Desorptionsprozess eine entscheidende Bedeutung zu. Die Desorption der Probe erfolgt durch Bestrahlung einer kleinen Fläche der Präparation ($\leq 150\mu\text{m}$) mit Laserlicht. Die Impulsdauer der Bestrahlung ist abhängig vom verwendeten Laser. Für moderne UV-Laser liegt sie im Bereich von ca. 1-20ns, für IR-Laser im Bereich von 50-150ns. Die Matrix dient zunächst der Absorption der Laserstrahlung. Durch die dem Matrix-Analyt-System zugeführte Energie gehen einige molekulare Schichten an der Oberfläche der bestrahlten Probe spontan in die Gasphase über. Dabei desorbieren neben den Matrixmolekülen auch die durch Kokristallisation während der Präparation in diese Schichten der Matrix eingebauten Analytmoleküle. Durch den hohen Matrixüberschuss werden intermolekulare Wechselwirkungen der Analytmoleküle untereinander weitgehend minimiert. Dies ermöglicht die Desorption einzelner, diskreter und großteils intakter, das heißt nicht-fragmentierter, Analytmoleküle.

Der Matrix wird weiterhin auch bei der Erzeugung der Ionen in der Gasphase eine entscheidende Rolle zugeschrieben [Olumee 1995]. Der Weg einer Analytionenbildung über eine primäre Photoionisation der Matrixmoleküle mit anschließendem Protonentransfer (bei der Bildung positiver Ionen) auf die Analytmoleküle ist zwar nicht eindeutig bewiesen, erscheint jedoch nach den bisher vorliegenden Daten wahrscheinlich.

Zur Veranschaulichung des Prinzips, welches der Trennung von Teilchen unterschiedlicher Massen zugrunde liegt, soll zunächst der einfachste Fall betrachtet werden: Zum Zeitpunkt der Desorption liegt an P1 die Spannung U_{IS1} an (P2 zunächst ohne Bedeutung). Die entstandenen Ionen werden durch die an P1 angelegte Spannung im elektrischen Feld zwischen P1 und G entsprechend ihrer Masse und Ladung unterschiedlich stark beschleunigt. In der Beschleunigungsphase gilt für die beschleunigende Kraft F :

$$F = m \cdot a = \frac{U_{IS1}}{d_a} \cdot z \cdot e \quad (1a)$$

m : Masse des Ions
 U_{IS1} : Beschleunigungsspannung an P1
 d_a : Beschleunigungsstrecke zwischen P1 und G
 z : Ladung des Ions
 e : Elementarladung ($1,60218 \cdot 10^{-19}\text{C}$)

Wobei für die Beschleunigung a in Formel (1a) gilt:

$$a = \frac{2 \cdot d_a}{t_a^2} \quad (1b)$$

t_a : Flugzeit während der Beschleunigung

Für die kinetische Energie E_{kin} eines so beschleunigten Ions gilt:

$$E_{kin} = U_{IS1} \cdot z \cdot e = \frac{m \cdot \left(\frac{d_d}{t_d} \right)^2}{2} \quad (2)$$

E_{kin} : kinetische Energie des beschleunigten Ions
 d_d : feldfreie Driftstrecke zwischen G und Detektor
 t_d : Flugzeit während der feldfreien Drift

Für die Gesamtflugzeit t_{gesamt} eines Ions gilt damit:

$$t_{gesamt} = t_a + t_d = \sqrt{\frac{m/z}{U_{IS1}}} \cdot \left[\left(d_a \cdot \sqrt{\frac{2}{e}} + d_d \cdot \sqrt{\frac{1}{2e}} \right) \right] \quad (3)$$

$$t_{gesamt} \propto \sqrt{\frac{m/z}{U_{IS1}}} \quad (4)$$

Somit hängt die Gesamtflugzeit bei konstanten Messparametern (d_a , d_d und U_{IS1}) für ein einfach geladenes Ion nur von der Masse ab. Die Kalibrierung des Massenspektrometers, also die Korrelation zwischen Masse und Flugzeit, kann so in einfacher Weise durch Messen der Flugzeiten von zwei oder mehr Substanzen bekannter Masse erfolgen. Die (linearen) Regression der registrierten Messdaten (Auftragung von Flugzeit gegen Masse) liefert die benötigte Kalibrierfunktion. Die berechneten Größen gelten in dieser Form exakt nur für eine Messung im „linearen Modus“ (Det(lin) in Abbildung 7).

Gemäß Gleichung (4) sollten bei einer Messung im linearen Modus alle Ionen mit identischen m/z -Werten exakt zur gleichen Zeit auf den Detektor Det(lin) treffen. Das resultierende Signal eines bestimmten Ions wäre damit theoretisch unendlich schmal. In der Praxis weist jedoch jedes, zu einem bestimmten m/z -Wert gehörige Signal eine endliche Breite auf. Die beobachtete Signalbreite ist dabei das Resultat einer Kombination von Energie-, Orts- und Zeitunschärfen. Orts- und Zeitschärfen haben ihre Ursache im wesentlichen im Desorptionsvorgang. Hier ist zum einen zu berücksichtigen, dass trotz eines nur sehr kurzzeitigen Laserpulses eine endliche Zeitspanne für Ionenbildung und Ionenextraktion existiert. Zum anderen werden die Ionen an verschiedenen Orten der sog. *plume* – das heißt, der „Teilchenwolke“ die durch die Desorption entsteht – gebildet. Beim Anschalten des elektrischen Feldes können somit Ionen gleicher m/z -Werte unterschiedlich lange Beschleunigungsstrecken durchlaufen – je nach Ort der Desorption und Ionenbildung. Durch Stöße der desorbierten Teilchen untereinander kann es unabhängig von den diskutierten Orts- und Zeitunschärfen auch zu einer

Verteilung der Anfangsenergien gleicher Analytmoleküle noch der Beschleunigungsphase kommen. In der Summe bewirken diese Faktoren eine starke Verbreiterung des beobachteten Peaks im linearen Modus und somit eine nur geringe Massenauflösung. Diese Auflösung ist bei TOF-Massenspektrometern definiert als:

$$R = \frac{m}{\Delta m} = \frac{m}{m(FWHM)} \quad (5)$$

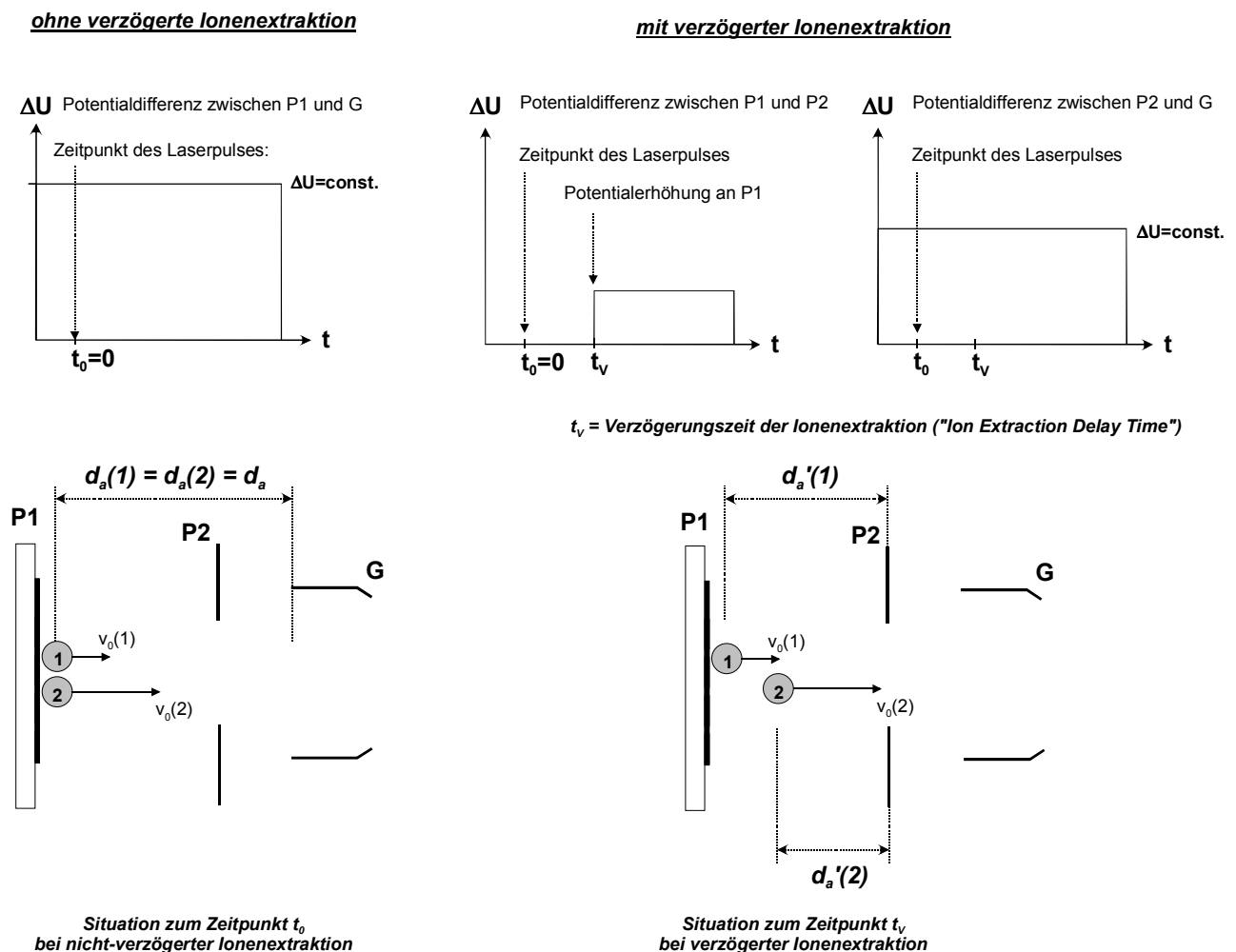
m : gemessene Masse im Peakmaximum
 $m(FWHM)$: Peakbreite auf halber Höhe des Peakmaximums

Energieunschärfen konnten durch Einführung der Techniken der zeitverzögerten Ionenextraktion (*delayed ion extraction*, abgekürzt „*de*“) [Wiley 1953, Brown 1995, Vestal 1995] und / oder die Messung im Reflektormodus (abgekürzt „*re*“) [Mamyrin 1973] entscheidend verbessert werden. Bei der Messung der Leiterpeptide zur Sequenzbestimmung liefert nur die *de*-MALDI-*re*-TOF-MS reproduzierbar die erforderliche Genauigkeit für eine Bestimmung der Aminosäuremassen.

Abbildung 8 zeigt das Prinzip der zeitverzögerten Ionenextraktion im Vergleich zur nicht-zeitverzögerten Ionenextraktion (nicht erklärte Bezeichnungen sind analog zu Abbildung 7). Die mit ① und ② bezeichneten Ionen sollen zwar gleiche Werte für m/z , aber unterschiedliche Anfangsgeschwindigkeiten besitzen. Bei der Ionenextraktion ohne „*de*“ (linke Seite in Abbildung 8) liegt das elektrische Beschleunigungsfeld zwischen P1 und G bereits zum Zeitpunkt der Ionenerzeugung (Laserpuls) an – P2 kommt hier keine Bedeutung zu. Ionen mit einer höheren Anfangsgeschwindigkeit $v_0(2)$ durchlaufen also bei eingeschaltetem elektrischen Feld die gleiche Beschleunigungsstrecke d_a wie Ionen mit einer geringeren Anfangsgeschwindigkeit $v_0(1)$. Die Aufnahme an kinetischer Beschleunigungsenergie ist damit für beide Ionen (① und ②) über d_a gleich. Ionen mit einer höheren Anfangsgeschwindigkeit erreichen den Detektor jedoch damit früher, als Ionen mit geringerer Anfangsgeschwindigkeit. Es kommt also zu unterschiedlichen Flugzeiten der gleichen Ionen (gleicher Wert für m/z) und folglich zu einer Peakverbreiterung für das Signal dieser Ionen. Bei der *delayed extraction* Technik (rechte Seite in Abbildung 8) befindet sich (im einfachsten Fall) P2 zunächst während der Desorption auf gleichem Potential befindet wie P1 (Target). Mit einer Verzögerungszeit t_V , die üblicherweise im Bereich einiger hundert Nanosekunden nach der Desorption und damit der Ionenerzeugung liegt, wird die Spannung von P1 durch einen computergesteuerten Pulsgenerator so erhöht, dass ein elektrisches Beschleunigungsfeld zwischen P1 und P2 entsteht. Ionen mit einer höheren Anfangsgeschwindigkeit $v_0(2)$ haben sich nach der Zeitspanne t_V bereits weiter vom Ort der Desorption entfernt. Wird

nun zum Zeitpunkt t_V das elektrische Feld angelegt, so ist die Strecke $d_a(2)$ zwischen P1 und P2, über die Ionen mit höherer Anfangsgeschwindigkeit $v_0(2)$ kinetische Beschleunigungsenergie aufnehmen können, geringer als für Ionen mit kleinerer Anfangsgeschwindigkeit $v_0(1)$. Ionen, die sich näher am Ort der Desorption befinden, nehmen also zwischen P1 und P2 mehr kinetische Energie auf (Beschleunigungsstrecke z.B. $d_a(1)$ für ①), als Ionen, die sich bereits weiter vom Ort der Desorption entfernt haben (Beschleunigungsstrecke z.B. $d_a(2)$ für ②). Die nachfolgende Beschleunigung, die beide Ionen dann noch zwischen P2 und G erfahren, ist identisch. Somit kann der durch unterschiedlichen Anfangsgeschwindigkeiten bedingte, kinetische Unterschied der Ionen ① und ② bei geeigneter Wahl der Verzögerungszeit t_V minimiert und eine höhere Massenauflösung erreicht werden.

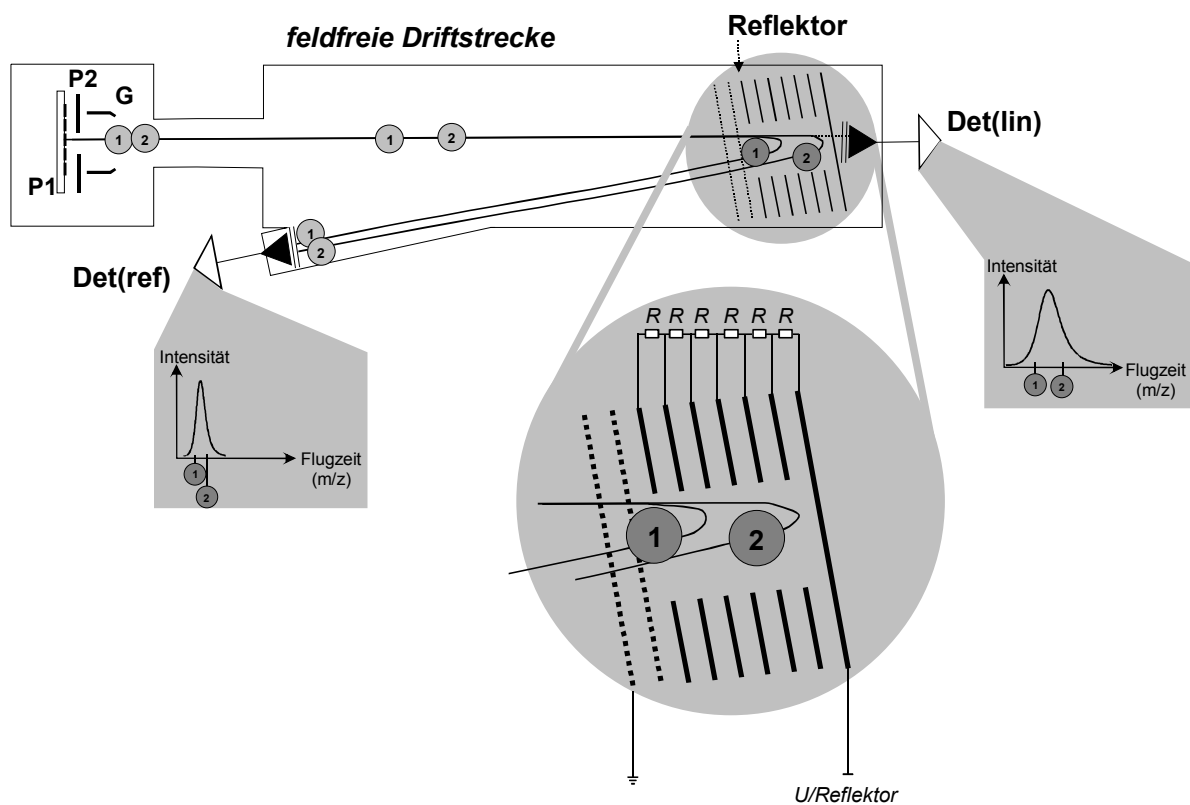
Abbildung 8: Prinzip der zeitverzögerten Ionenextraktion (*delayed ion extraction*); nicht erklärte Bezeichnungen siehe Abbildung 7



Eine weitere Möglichkeit, um die durch den Desorptionsprozess bedingten Energieunschärfen zu minimieren und damit die Massenauflösung zu erhöhen, ist durch die Verwendung des

bereits in Abbildung 7 gezeigten „Reflektors“ gegeben. Abbildung 9 veranschaulicht das Funktionsprinzip des Reflektors. Die mit ① und ② bezeichneten Ionen sollen auch hier wieder das gleiche m/z -Verhältnis, jedoch unterschiedliche Startenergien aufweisen. Der Reflektor ist in seinem Aufbau vergrößert dargestellt.

Abbildung 9: Funktionsprinzip eines Reflektors zu Erhöhung der Massenauflösung



Zur Vereinfachung erfolgen die Betrachtungen ohne *delayed-extraction*. Erfolgt die MALDI-MS im linearen Modus, so liegt am Reflektor keine Spannung an und die Ionen werden mit dem linearen Detektor Det(lin) erfasst (gestrichelte Flugbahn). Im linearen Modus spiegelt sich die Anfangsenergieverteilung von Ionen gleicher m/z -Werte direkt in der Peakbreite wider. Je größer die Anfangsenergieverteilung ist, desto breiter erscheinen die Peaks im linearen Modus.

Im Reflektormodus wird über die Spannung „*U/Reflektor*“ sowie die Widerstände *R* ein elektrischer Feldgradient erzeugt, der eine zur Flugrichtung entgegengesetzte Beschleunigung bewirkt, sobald die Ionen in den Reflektor eindringen. Das elektrische Feld, das die Ionen erfahren ist umso größer, je weiter die Ionen in den Reflektor eindringen. Ionen eines definierten Wertes für m/z mit größerer kinetischer Energie, wie z.B. das Ion ②, werden weiter in den Reflektor eindringen können, bevor sie durch das Gegenfeld in Richtung des Reflektordetektors Det(ref) zurückbeschleunigt werden, als Ionen mit geringerer kinetischer

Energie aber des gleichen Wertes für m/z . Bei optimierten Parametern für Geometrie und Spannung des Reflektors kann der Unterschied in den Startenergien stark verringert und eine zeitliche Fokussierung am Ort des Reflektordetektors erreicht werden. Die minimierte Differenz der Ankunftszeiten der Ionen ① und ② am Ort des Detektors spiegelt sich in einer erhöhten Auflösung wieder.

1.2.2 Spezielle Aspekte beim enzymatischen Leitersequenzieren

1.2.2.1 Spektreninterpretation, Massengenauigkeit und isobare Aminosäuren

Da beim MALDI Prozess überwiegend einfach geladene Molekülionen der Peptide gebildet werden, kann aus den Differenzen der gemessenen m/z -Werte der Leiterpeptide direkt die Masse der abgespaltenen Aminosäuren abgelesen werden. Für die Spektreninterpretation ist dabei die absolute Massengenauigkeit nicht entscheidend. Für die Massenanalyse erfolgt eine Kalibrierung in der Regel extern, d.h. die Substanzen zur Errechnung der Kalibrierfunktion befinden sich nicht direkt in der Analysenmischung. Meist nimmt die Genauigkeit zu, je mehr solcher Substanzen zur Kalibrierung herangezogen werden, da eine exakte Linearität der Kalibrierfunktion nicht immer gegeben ist. Die Genauigkeit einer externen Kalibrierung hängt dann jedoch oft auch davon ab, ob die Messung der Analyten unter weitgehend analogen Bedingungen, z.B. Laserintensität oder Analyt-Matrix-Verhältnis, erfolgen kann. Im optimalen Fall werden so nach externer Kalibrierung Massengenauigkeiten in Bereich bis 10ppm erreicht. Für die Proteinidentifizierung über *mass-fingerprints* in Datenbanken ist diese Genauigkeit oft nicht ausreichend. Hier kann nur eine interne Kalibrierung die notwendige Genauigkeit im unteren ppm-Bereich (1-10ppm) liefern. In der Praxis werden meist Signale von Autolyseprodukte der verwendeten Spaltenzyme zur Kalibrierung herangezogen. Bei Endoproteinasespaltungen mit Trypsin lassen sich beispielsweise sehr gut die Autolyseprodukte des Enzyms mit $[M+H]^+_{\text{monoisotopisch}} = 805,417 \text{ Da}$ (Fragment 112-119) und $[M+H]^+_{\text{monoisotopisch}} = 2163,057 \text{ Da}$ (Fragment 70-89) zur Kalibrierung heranziehen. Das nachträgliche Zumischen eines Standards zu den Analyten als Form der internen Kalibrierung bringt normalerweise nicht das gleiche Maß an Genauigkeit, was zumeist an den stark unterschiedlichen Verhältnissen Analyt-Matrix und Kalibrand-Matrix liegt. Bei Proben unbekannter Konzentration ist dieses Problem nicht umgebar.

Für die Leitersequenzierung ist die Massengenauigkeit zur Identifizierung der meisten Aminosäuren im Spektrum zunächst unkritisch. Selbst bei externer Kalibrierung stimmen die Differenzen der Leiterpeptidmassen bis auf mindestens 100ppm genau mit den Aminosäure-

massen überein. Diese Genauigkeit reicht für alle 20 Standardamino­säuren, bis auf folgende Paare zur Identifizierung (Massenangaben monoisotopisch):

❖ **Leucin (Leu)** → 113,08406 Da / **Isoleucin (Ile)** → 113,08406 Da

❖ **Glutamin (Gln)** → 128,05858 Da / **Lysin (Lys)** → 128,09496 Da

Leu und Ile sind isobar – hier ist eine Unterscheidung aus dem Massenspektrum allein nicht möglich. Bei Gln und Lys hingegen beträgt die Massendifferenz 36,38 mDa. Bei interner Kalibrierung wäre hier also eine Unterscheidung durchaus möglich.

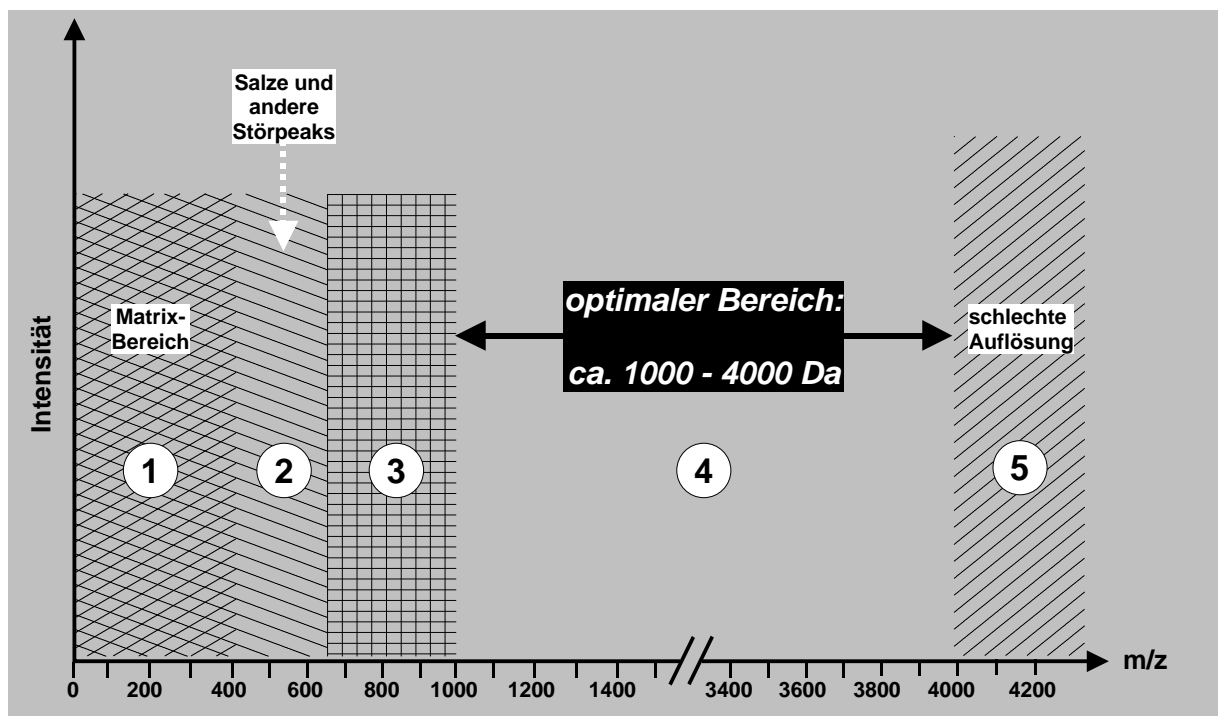
1.2.2.2 *Der Einfluss von Puffersalzen*

Ein weiterer Vorteil der MALDI-MS Analyse bei der Leitersequenzierung im Vergleich zu anderen massenspektrometrischen Detektionsmethoden, wie z.B. der ESI-MS, ist die Toleranz gegenüber relativ hohen Puffersalz-Konzentrationen. Enzyme für die Sequenzierung werden in der Regel bei Pufferkonzentration angewendet, die im millimolaren Bereich liegen. Bei geeigneter Wahl der Puffersubstanz treten mit diesen Pufferkonzentrationen bei der MALDI-MS zumeist noch keine Probleme auf. Grundsätzlich sollte jedoch auch bei den enzymatischen Sequenzierungen die Pufferkonzentration nur gerade so hoch gewählt werden, um eine ausreichende Pufferkapazität zu gewährleisten (konstante/reproduzierbare Bedingungen während der Sequenzierung). Soweit möglich, sollten als Puffersubstanzen sog. „flüchtige“ Puffer, wie z.B. Ammoniumcarbonat, Ammoniumbicarbonat oder Ammoniumacetat verwendet werden. Diese Puffer zerfallen im Massenspektrometer zu unproblematischen, flüchtigen Komponenten wie Ammoniak, Kohlendioxid, Essigsäure und Wasser bzw. neigen weniger zur Bildung von Clustern oder Addukten, die im Spektrum stören könnten. Insbesondere Natriumsalze führen meist zur Bildung eines oder mehrerer $[M+xNa]^+$ -Ionen, statt zur Bildung von $[M+H]^+$. x entspricht im wesentlichen der Zahl saurer Gruppen im Peptid. Je größer x , desto stärker verteilt sich die Intensität des einzelnen $[M+H]^+$ -Ions auf mehrere Natriumaddukte. Die maximale Signalintensität eines Leiterpeptids und damit auch die Empfindlichkeit der Sequenzierung können so deutlich abnehmen.

1.2.2.3 *Analytisch nutzbarer Bereich des Massenspektrums*

Bei der Kopplung einer Leitersequenzierung mit MALDI-TOF kann man das Massenspektrum in mindestens fünf Bereiche unterteilen, die sich bezüglich ihrer Eignung für die Sequenzierung unterscheiden (Abbildung 10).

Abbildung 10: Eignung unterschiedlicher Massenbereiche im MALDI-MS für die Leitersequenzierung



Die Positionen der eingezeichneten Übergänge für jeden Bereich dienen lediglich als Anhaltspunkt. Sie variieren in Abhängigkeit der verwendeten Matrix, des gewählten Puffersystems für die Enzyme und dem zu sequenzierenden Peptid.

Der Matrixbereich ① erstreckt sich je nach verwendeter Matrix bis zwischen 300 und 400 Da. Da die Matrix in einem 10^3 - bis 10^5 -fachen molaren Überschuss zu den Analyten vorliegt, käme es durch den Ionenstrom der Matrixionen beim Auftreffen auf den Detektor zu einer Sättigung, und damit zu einer starken Verringerung der Empfindlichkeit für nachfolgende Analytionen. Ionen im niedermolekularen Bereich und somit vor allem die Matrixionen werden deshalb durch eine kurzzeitig senkrecht zur Flugrichtung angelegte Spannung am Deflektor (siehe Abbildung 7: D) von der linearen Flugbahn abgelenkt und können somit nicht mehr auf den Detektor auftreffen. Wenn alle Matrixionen den Deflektor passiert haben wird die Ablenkspannung ausgeschaltet und die nachfolgenden langsameren (schwereren) Ionen können detektiert werden. Bei einer Deflektionszeit, die der Masse der Matrix entspricht (Gleichung 4) sind somit allerdings auch kleinere Leiterpeptide mit weniger als 3-5 Aminosäuren nicht mehr analytisch fassbar. Während bei einer enzymatischen Sequenzierung also durchaus ein kompletter Abbau des Peptids bis zu den einzelnen Aminosäuren erfolgen kann, endet die detektierbare Sequenz meist spätestens beim Tetrapeptid. Leider sind auch die abgespaltenen Aminosäuren nicht detektierbar. Selbst wenn auf die Deflektion der Matrixionen verzichtet wird, ist die Menge der abgespaltenen Aminosäuren und der

Leiterpeptide bei Sequenzierungen im Pikomol-Bereich so gering, dass deren Signale durch die Matrixsignale vollkommen unterdrückt werden.

Auch im Bereich ② treten bei enzymatischen Sequenzierungen mehr oder weniger intensive, störende Signale von Ionen oder Ionencuster der Puffersalze auf. Der Bereich ist daher für die Sequenzierung nicht grundsätzlich unzugänglich wie der Bereich ①. Ob der Bereich genutzt werden kann, hängt auch vom sequenzierten Peptid ab. Ionisierbarkeit und Menge des Peptids einerseits, sowie der verschiedenen Störsubstanzen andererseits entscheiden darüber, ob eine Unterdrückung der Leiterpeptidionen stattfindet oder nicht. Da jedoch auch die molaren Konzentration der Puffersubstanzen in Regel um den Faktor 10^3 höher liegen, als die Konzentration der Analytmoleküle, wird in den meisten Fällen eine Suppression der Leiterpeptidsignale eintreten. Um eine potentielle Verringerung der Empfindlichkeit bei der Detektion der Leiterpeptide zu vermeiden empfiehlt es sich, die Spannung am Deflektor entsprechend länger anzuschalten, so das zusätzlich zu den Matrixionen auch störenden Ionen im Bereich ② (bis ca. 500-600 Da) noch abgelenkt werden.

Aus dem bisher Gesagten ergibt sich, dass eine Sequenzierung kleiner Peptide mit bis zu sechs oder sieben Aminosäuren nicht oder nur begrenzt möglich ist. Bei Hexa- oder Heptapeptiden, die dabei an der oberen Grenze des Bereichs ② liegen, kann somit maximal die Abspaltung von zwei bis drei Aminosäuren beobachtet werden. Generell kann aber auch von größeren Peptiden nie die komplette Sequenzinformation erhalten werden, wenn das Peptid nur von einem Terminus aus – also nur C- oder N-terminal – sequenziert wird. Die Information über die letzten drei bis sieben Aminosäuren bei C-terminaler Sequenzierung muss also durch zusätzliche N-terminale Sequenzierung bestimmt werden und umgekehrt. Aus dem Vorangegangenen folgt, dass theoretisch erst für Peptide mit einer Masse von mehr als ca. 800 – 1000 Da die gesamte Sequenz erhalten werden kann, wenn von beiden Termini her sequenziert wird. Daher erlauben erst Peptide mit Massen im Bereich ③ eine komplette Sequenzanalyse.

Bereich ④ stellt mit ca. 1000 – 4000 Da den optimalen Massenbereich für die enzymatische Leitersequenzierung gekoppelt mit MALDI-TOF-MS dar. Die Auflösung moderner *de-MALDI-re-TOF* Massenspektrometer erlaubt hier für nahezu alle Peptide eine monoisotopische Auflösung der entsprechenden Leiterpeptidesignale. Dies ist eine notwendige Voraussetzung, um die Massen der Leiterpeptide und somit auch die Massendifferenzen mit ausreichend hoher Genauigkeit zu bestimmen. Die absolute Richtigkeit der gemessenen Peptidmassen ist zwar für die Bestimmung der Massendifferenzen theoretisch nicht entscheidend, wohl aber die Präzision. Diese Präzision

ist für monoisotopische Massen im unteren Massenbereich am höchsten. Mit abnehmender Präzision, d.h. erhöhtem Δm , nimmt auch $\Delta(\Delta m)$ zu. Die obere Grenze von 4000 Da ist hier willkürlich gewählt, denn zum Teil können auch Signale von Peptiden mit Massen um 5000 Da noch monoisotopisch aufgelöst werden. Neben den Spannungsparametern für Beschleunigung, Reflektor und Detektor sowie der Beschleunigungsverzögerung (*extraction delay*), spielt auch die Peptidmenge eine entscheidende Rolle dafür, ob sich ein Peptidsignal in die einzelnen Isotopenpeaks auflösen lässt. Für die üblicherweise verwendeten Spannungsparameter im Peptidbereich sind monoisotopische Auflösungen von ca. 3500 Da bis max. 4000 Da als realistisch anzusehen – zumindest im Regelfall. Eine Optimierung für Einzelfälle über 4000 Da ist zwar teilweise möglich, entspricht jedoch nicht der Aufgabenstellung dieser Arbeit, wonach eine möglichst universelle Methode erarbeitet werden soll.

Im Massenbereich ⑤ ist die Ermittlung der Sequenz durch die Leiterpeptidsignale zwar immer noch möglich, allerdings nimmt mit zunehmender Peptidmasse Auflösung der Signale ab und damit auch die Genauigkeit der Massendifferenzen. Insbesondere der Übergang von der Messung monoisotopischer Signale zur Messung von Durchschnittsmassen ist bei der Differenzbildung zweier Peaks kritisch, da sich mit zunehmender Masse Durchschnittsmasse und monoisotopische Masse der Peptide sehr deutlich unterscheiden. So unterscheiden sich zum Beispiel bei Peptiden mit Massen zwischen 3000 und 4000 Da die Durchschnittsmasse und die monoisotopische Masse um ca. 2,0 bis 2,7 Da. Ein weiteres Problem in Bereich ⑤ ergibt sich durch mögliche Fragmentierungen der Leiterpeptide. Vor allem metastabile Fragmentierungen auf der feldfreien Driftstrecke finden je nach Peptid und Lasereinstellung immer in unterschiedlich starkem Ausmaß statt und bilden auch die Grundlage für Strukturermittlungen mittels *post source decay*. Charakteristisch für die Fragmentionen ist dabei deren geringere Auflösung im Vergleich zu Ionen nicht-fragmentierter Peptide. Somit lassen sich gerade Leiterpeptidionen geringer Intensität von den ebenfalls meist intensitätsschwächeren Fragmentionen im Massenbereich ④ sehr gut unterscheiden, wodurch die Spektreninterpretation stark vereinfacht wird. Geht die Isotopenauflösung eines Peptidsignals jedoch – wie unter Umständen im Bereich ⑤ des Massenspektrums – verloren, so erschwert dies auch die Interpretation der Leiterspektren. Die Unterscheidung der Leiterpeptidionen zu Fragmentionen ist daher auch ein Grund dafür, weshalb nur die hochauflösende *de*-MALDI-*re*-TOF-MS für die Leitersequenzierung geeignet erscheint. Von größeren Peptiden kann somit letztlich nur derjenige Sequenzteil mit hoher Genauigkeit eindeutig bestimmt werden, dessen Leiterpeptide im Bereich ④ liegen.

1.3 Praktische Ausführungsformen der Leitersequenzierung

1.3.1 Spaltung in Lösung, *on-target* oder auf der Membran

Für die konkrete Durchführung einer enzymatischen Leitersequenzierung sind mehrere Ausführungsformen möglich. Drei Varianten, die auch in dieser Arbeit untersucht wurden, sollen im Folgenden näher beschrieben werden:

Variante A: Sequenzierung in Lösung

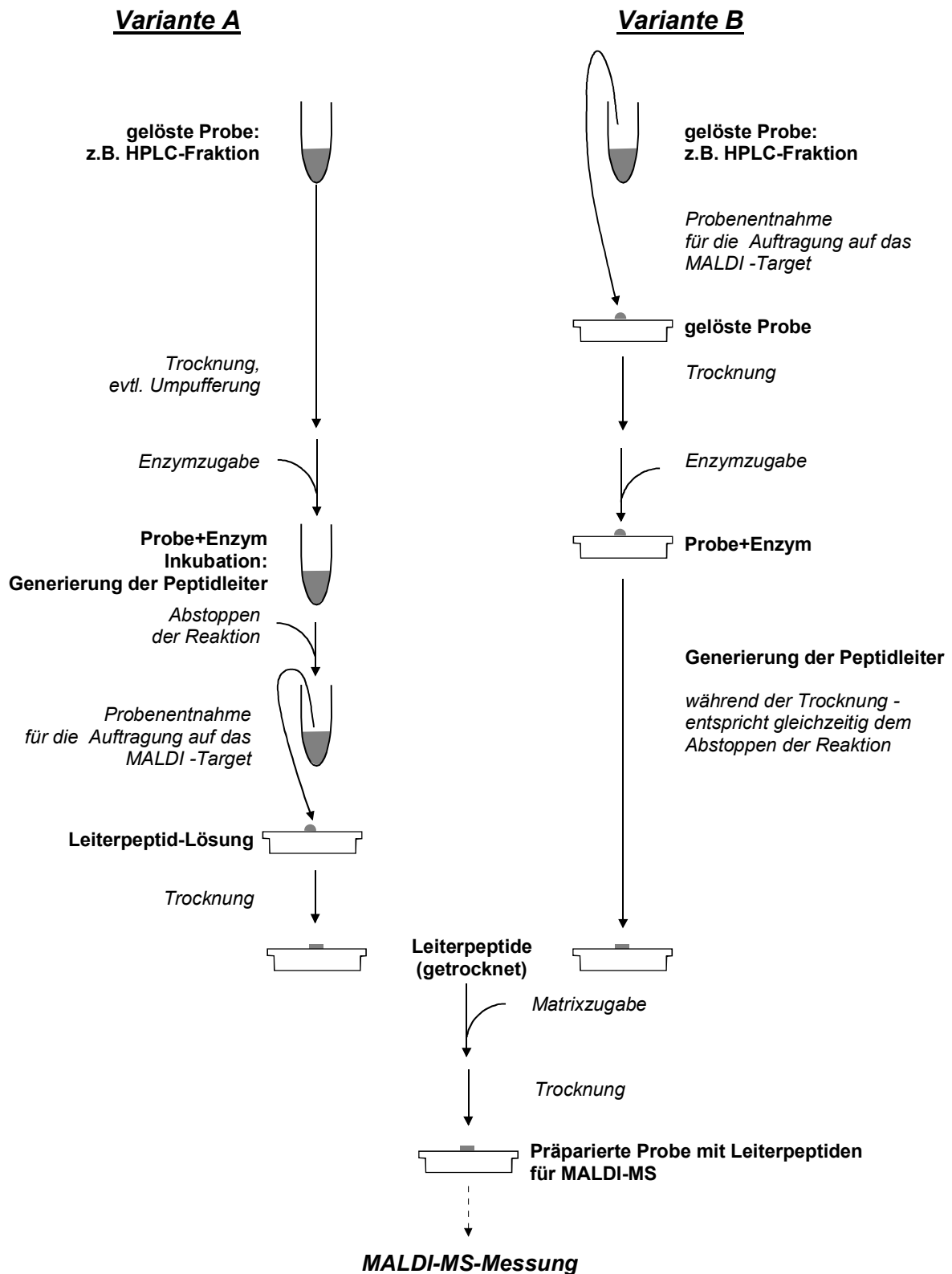
Variante B: Sequenzierung auf dem MALDI-Probenträger (*on-target*)

Variante C: Sequenzierung auf einer PVDF-Membran

Abbildung 11 (folgende Seite) fasst die Arbeitsabläufe der Varianten A und B schematisch zusammen. Variante A entspricht auch der klassischen Vorgehensweise einer enzymatischen Sequenzierung, bei welcher die Aminosäuren oder deren Derivate zur Detektion verwendet werden. Zu dem zu sequenzierenden Peptid im Spaltpuffer wird das Enzym zugegeben und anschließend für eine bestimmte Zeit inkubiert, wie z.B. in [Thiede 1995] oder [Holland 1996] beschrieben. Danach wird die Reaktion gestoppt, was zumeist durch Zugabe von Säure geschieht (pH-Wert auf ca. 2-3 absenken). Anschließend wird ein Aliquot der Spalltlösung entnommen und für die MALDI-Analyse, wie unter 1.2.1 beschrieben, präpariert. Die pH-Absenkung kann auch direkt durch Mischen eines Aliquots des Spaltansatzes mit Matrixlösung erfolgen.

Variante B wird in der Literatur allgemein als *on-target* Sequenzierung bezeichnet [Patterson 1995]. Alle Arbeitsschritte der Sequenzierung erfolgen direkt auf dem metallischen Träger für die MALDI-Proben. Bei der *on-target* Sequenzierung werden nacheinander Probe, Enzymlösung und Matrix auf den Probenträger pipettiert. Die verwendeten Volumina bewegen sich, bedingt durch die Probenspotgröße, im Bereich von ca. 1 µl und weniger. Eine direkte Sequenzierung auf dem Probenträger hat mehrere Vorteile gegenüber einer Spaltung in Lösung. Zum einen findet beim Arbeiten mit sehr kleinen Probenvolumina im Sub-µl-Bereich keine Verdünnung der Probe statt, auch nicht bei der Anwendung sequenzieller Exopeptidasekombinationen, sofern man die einzelnen Lösungen immer wieder auf dem Probenspot eintrocknen lässt. Die gesamte, für das Sequenzierungsexperiment eingesetzte Probenmenge steht somit für eine einzige MALDI-MS Detektion zur Verfügung. Des weiteren können Verluste, z.B. durch Adsorption der Probe an Gefäßwände oder Transferverluste durch Umpipettieren der Probenlösungen vermieden werden. Gerade bei längerem Arbeiten mit hydrophoberen Peptiden besteht in wässrigen Lösungen die Gefahr von Adsorptionsverlusten an den Polypropylenwänden der Probengefäße.

Abbildung 11: Arbeitsschritte bei Sequenzierung in Lösung (A) und auf dem MALDI-Probenträger (B)



Ein zusätzlicher Aspekt, der für die *on-target* Variante spricht, ist die wesentlich einfachere Umpufferung der HPLC-getrennten internen Peptidfragmente eines Proteins. Übliche

Trennungen solcher Peptidgemische werden auf C₁₈-Umkehrphasen mittels eines Gradienten aus Wasser/TFA und Acetonitril/TFA durchgeführt. Die eluierenden Peptide befinden sich also zunächst in einem für die Exopeptidasesequenzierung inkompatiblen Lösungsmittel. In der Regel beträgt der Acetonitrilgehalt der Peptidfraktionen 10 – 70% und der pH-Wert liegt in 0,05 – 0,1% TFA bei ca. pH 2. Die Umpufferung der kleinen Elutionsvolumina erfolgt häufig über Gefriertrocknung oder durch Vakuumzentrifugation (*speed-vac*). Anschließend wird das Peptid in einem zur Exopeptidasespaltung kompatiblen Puffer wieder aufgenommen. Diese Arbeitsschritte, die für eine Sequenzierung in Lösung erforderlich sind, sind nicht nur zeitaufwendig, sondern zumeist auch mit sehr großen Adsorptionsverlusten verbunden. Insbesondere das Einengen bis zur Trockene ist ein sehr kritischer Schritt. Die *on-target* Spaltung erlaubt hingegen durch ein offenes System nach der Auftragung der Peptidfraktion auf den Probenträger eine rasche Verflüchtigung des HPLC-Elutionsmittels ohne zusätzlichen Arbeits- oder Geräteaufwand. Bei einem Probenvolumen von ca. 1 µl erfolgt eine Verflüchtigung des organisch-wässrigen Lösungsmittels der HPLC bei Acetonitril als organischer Komponente je nach Zusammensetzung und Umgebungstemperatur meist innerhalb weniger Minuten. Auch im Hinblick auf eine Automatisierung der Sequenzierungstechnik erscheint die *on-target* Variante so besser geeignet. Allerdings ist bei der *on-target* Sequenzierung die Kontrolle der Reaktionsbedingungen wesentlich kritischer als bei einer Sequenzierung in Lösung. Obwohl sich die Reaktionstemperatur über den Probenträgerträger aus Metall und das geringe Volumen des Spaltansatzes recht gut kontrollieren lässt, erschwert das offene System die Einhaltung exakt reproduzierbarer Bedingungen. Insbesondere Schwankungen in der Luftfeuchtigkeit und wechselnde Luftbewegungen über dem Probenträger können zu stark variierenden Trocknungs- und damit auch Spaltzeiten des Ansatzes führen.

Die Spaltung auf einer Membran nach Variante C ist, ebenso wie die *on-target* Spaltung, besonders bei Kopplung mit der Mikro-HPLC oder Kapillar-HPLC interessant, da hier die Fraktionsvolumina so gering sind, dass ein „einfaches Fraktionssammeln“ praktisch nicht mehr möglich ist. Das gesamte HPLC-Eluat wird hier über eine Kapillare direkt auf einer hydrophoben Polyvinylidenfluorid (PVDF) – Membran verteilt und somit gleichzeitig „fixiert“ [Eckerskorn 1997]. Die Kapillare bewegt sich dabei während der Elution mit konstanter Geschwindigkeit über den Membranstreifen und setzt in regelmäßigen, kurzen Intervallen das Eluat auf der Membran ab. Damit befindet sich an jeder Stelle des Membranstreifens am Ende der Trennung eine Peptidfraktion, deren Korrelation zum aufgezeichneten Chromatogramm in einfacher Weise möglich ist. Für Peptide, die über

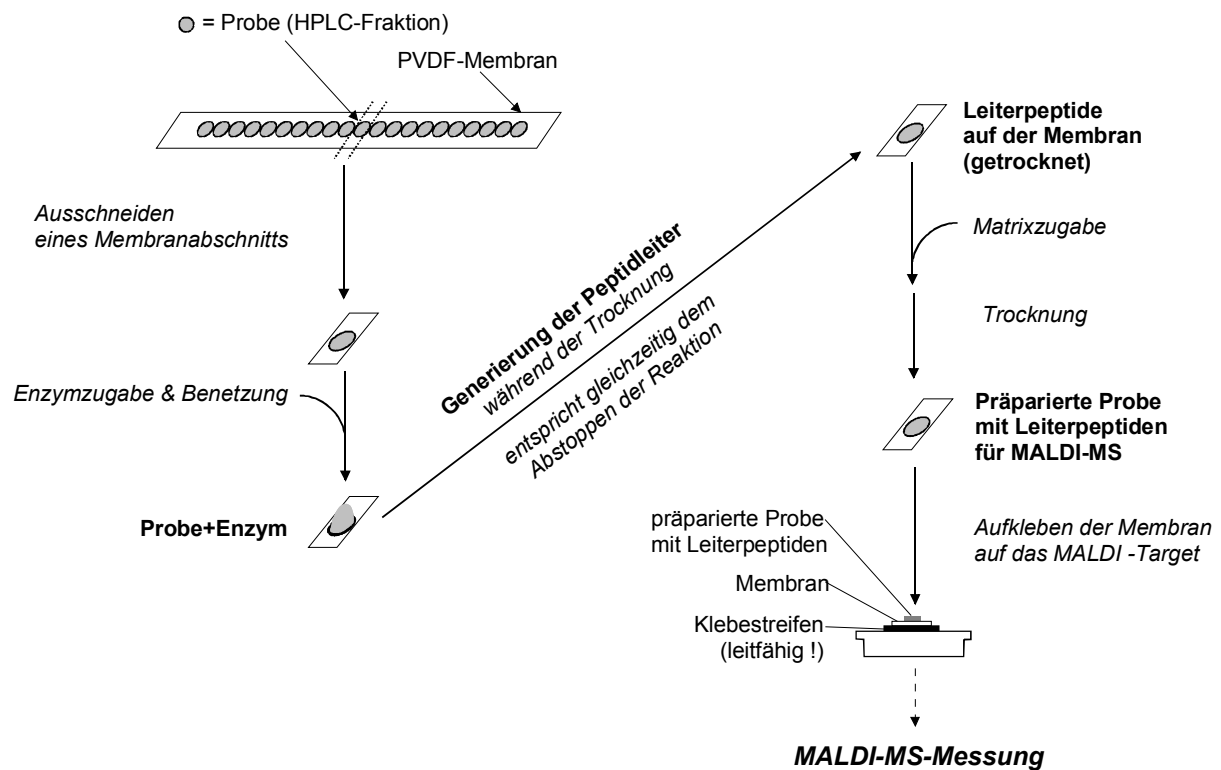
hydrophobe Wechselwirkungen auf und in der Membran fixiert sind, kann eine Sequenzierung durch Inkubation der Membran in einer Lösung der Exopeptidase erfolgen. Alternativ können interessierenden „Fraktionen“ aus der Membran ausgeschnitten und einzeln gespalten werden. Inwiefern eine solche Spaltung auf der Membran erfolgreich ist, wird jedoch sehr stark von der Benetzungsfähigkeit der Membran mit der Enzymlösung abhängen. Mit wässrigen Lösungen findet eine Benetzung hydrophober PVDF-Membran kaum statt und somit ist eine Reaktion der Peptidase mit dem Peptid kaum möglich. Durch die Zugabe eines geringen Prozentsatzes an organischem Modifier wie Methanol, Ethanol oder Acetonitril zur Enzymlösung kann die Benetzbarkeit stark erhöht werden. Der organische Anteil im Spaltungspuffer muss dabei nicht zwangsläufig zu einer Inaktivierung der Peptidase führen. Geringe Gehalte an Acetonitril zum Beispiel können je nach Enzym durchaus tolerierbar sein und ganz im Gegenteil sogar zu einer Steigerung der Enzymaktivität führen. Tabelle 5 zeigt dies anhand der gemessenen Enzymaktivitäten von Carboxypeptidase C (EC 3.4.16.1) aus Hefe und *penicillium janthinellum*. Alle Werte wurden auf eine Enzymaktivität von 100% ohne Zusatz von Acetonitril normiert.

Tabelle 5: Relative Enzymaktivitäten von Carboxypeptidase C bei verschiedenen Acetonitrilgehalten

Acetonitrilgehalt	Enzymaktivität von Carboxypeptidase C (EC 3.4.16.1) aus	
	Hefe	<i>penicillium janthinellum</i>
1% (v/v)	122%	108%
5% (v/v)	157%	47%
10% (v/v)	161%	70%

Ein aktivitätssteigernder Effekt wird dabei im allgemeinen auf die denaturierende Wirkung des organischen Lösungsmittels zurückgeführt, welche das Peptid für die Spaltung besser zugänglich macht. Organische Solvenzien als Benetzungszusatz sind jedoch nicht unbedenklich bei der Inkubation, wenn sich auf dem so behandelten Membranstreifen mehrere Fraktionen nebeneinander befinden. Je nach Stärke der hydrophoben Wechselwirkung zwischen Peptid und Membran kann durch das organische Solvens eine Auswaschung aus der Membran erfolgen. Für die bei der Sequenzierung generierten Leiterpeptide ändern sich zudem die hydrophoben Eigenschaften mit dem Grad der Abspaltung einzelner Aminosäuren. Auch die zum Schluss aufgetragene Matrix befindet sich in Lösungen mit mindestens 30% Acetonitril. Auch hier kann es zu einer Auswaschung der Leiterpeptide aus der Membran oder einem Verlust der Ortsauflösung kommen.

Abbildung 12: Arbeitsschritte bei Sequenzierung auf einer Membran

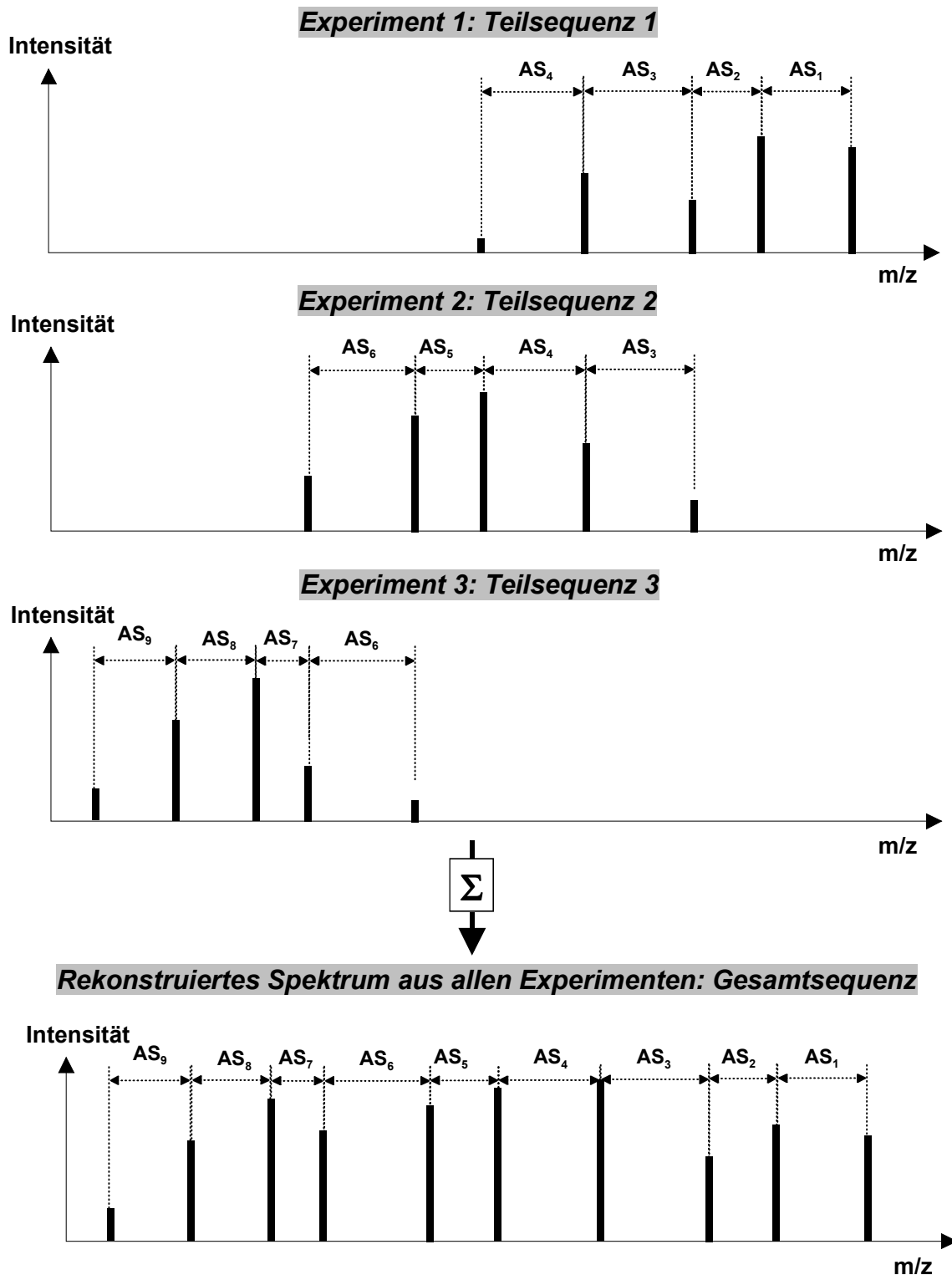


Aus praktischen Gesichtspunkten erscheint daher die Sequenzierung von Peptiden, die auf einer Membran gesammelt wurden, nur dann möglich, wenn wie in Abbildung 12 dargestellt vorgegangen wird. Die einzelnen Fraktionen auf der Membran werden zunächst ausgeschnitten und dann getrennt sequenziert werden.

1.3.2 Zeit- und Konzentrationsabhängige Spaltungen

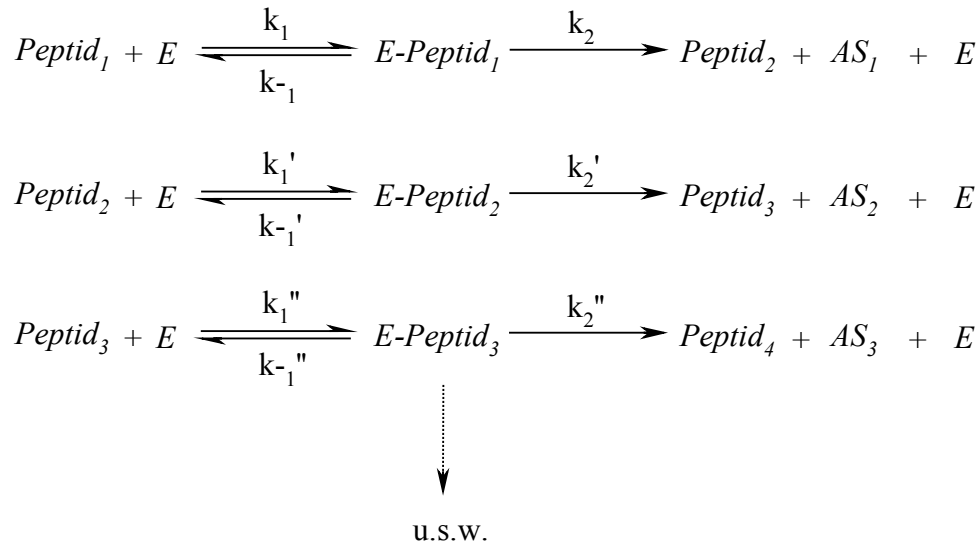
Die Abspaltungsgeschwindigkeit ist, wie in Abbildung 5 dargestellt, von einer Vielzahl miteinander vernetzter Parameter abhängig. Hat man hier einen Parametersatz gefunden, der eine gute kinetische Kontrolle der Aminosäureabspaltung ermöglicht, so wird es dennoch bei den meisten Sequenzierungen nicht gelingen, für ein Peptid mit nur einem einzigen Experiment eine komplette Leitersequenz über den gesamten analytisch nutzbaren Bereich des Massenspektrums zu erhalten – besonders bei längeren Peptiden. Mit dem Fortschreiten der Sequenzierung werden größere Leiterpeptide in zunehmenden Umfang abgebaut und verschwinden zum Teil auch ganz. Daher ist es vielmehr nötig, durch eine Serie geeigneter Experimente die Kinetik der Aminosäureabspaltung so zu verfolgen, dass jeweils überlappende Sequenzabschnitte erhalten werden. Aus diesen lässt sich letztlich die Gesamtsequenz rekonstruieren. Abbildung 13 zeigt dieses Vorgehen schematisch.

Abbildung 13: Rekonstruktion der Gesamtsequenz aus Teilsequenzen bei der Leitersequenzierung



Möglichkeiten, die sukzessive Aminosäureabspaltung so zu steuern, dass im Experiment solche sich ergänzenden Teilsequenzen wie in Abbildung 13 zu erhalten sind, ergeben sich durch Variation von Spaltungszeit und / oder Enzymkonzentration [Patterson 1995].

Allgemein können für die einzelnen Abbauschritte Reaktionsgleichungen aufgestellt werden, wie sie auch bei der Ableitung der Michaelis-Menten-Konstante und der Michaelis-Gleichung formuliert werden:



*Peptid*₁, *Peptid*₂, ...: Ausgangspeptid und folgende Leiterpeptide
E: Enzym (Exopeptidase)
*E-Peptid*₁, *E-Peptid*₂, ...: Enzym-Substrat-Komplexe
*AS*₁, *AS*₂, ...: abgespaltene Aminosäuren
*k*₁, *k*₋₁, *k*₂: Geschwindigkeitskonstanten

Geht man bei den einzelnen enzymatischen Abbauschritten vereinfacht von Bedingungen aus, die auch der Ableitung der Michaelis-Gleichung zugrunde gelegt werden [Voet 1995], so folgt für den Abbau von *Peptid*₁ im ersten Reaktionsschritt:

$$v = \frac{d[\text{Peptid}_2]}{dt} = k_2 \cdot V_{\max} \cdot \frac{[\text{Peptid}_1]}{K_M + [\text{Peptid}_1]} \quad (6)$$

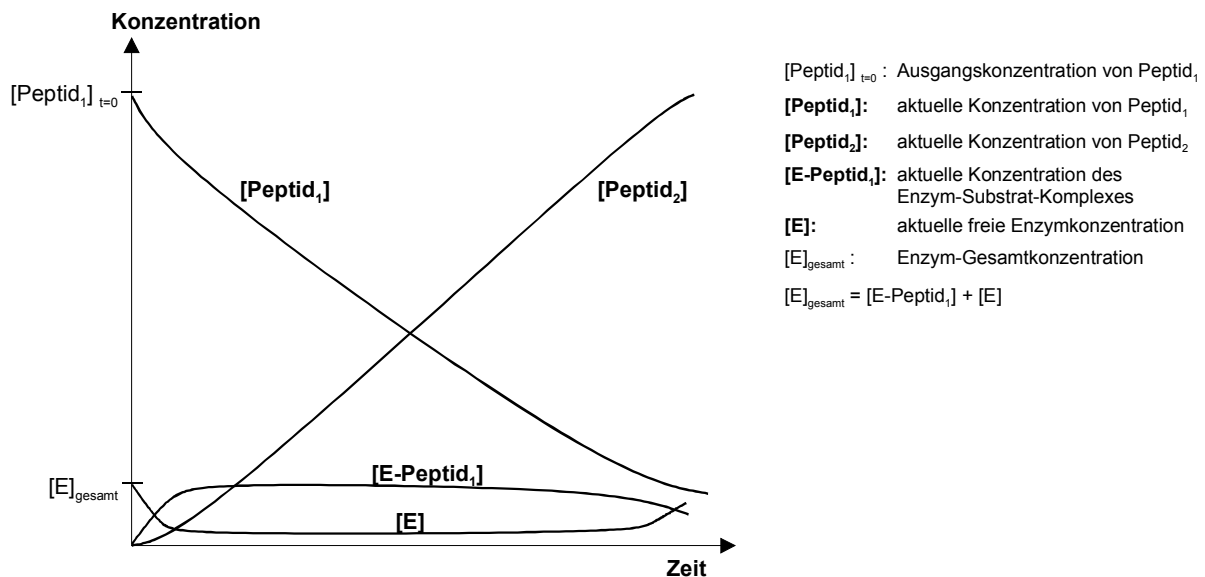
$$\text{mit: } K_M = \frac{k_{-1} + k_2}{k_1} : \quad \text{Michaelis-Konstante}$$

$$V_{\max} = k_2 \cdot [E_{\text{gesamt}}] : \text{maximale Reaktionsgeschwindigkeit} \quad (7)$$

v: Reaktionsgeschwindigkeit
[E_{gesamt}]: gesamte Enzymkonzentration

mit der Bildung des um eine Aminosäure verkürzten Leiterpeptids wird im vergleichbaren Maß das Ausgangspeptid in seiner Konzentration abnehmen. Die Konzentrationsverläufe der einzelnen Reaktionsteilnehmer für diesen einzigen Reaktionsschritt stellen sich dann wie in Abbildung 14 dar.

Abbildung 14: Konzentrationsverläufe der Reaktionsteilnehmer einer enzymatischen Aminosäureabspaltung bei Betrachtung eines einzelnen, isolierten Abspaltungsschritts



Grafik modifiziert für die Leitersequenzierung nach [Voet 1995]

Bei der Leitersequenzierung ist nun jedes Peptidprodukt einer Aminosäureabspaltung, also z.B. auch *Peptid₂*, gleichzeitig wieder Substrat für den nächsten Abspaltungsschritt. Die gesamte Beschreibung der kinetischen Verhältnisse auf mathematischer Basis ist hier daher nicht ganz so einfach wie bei der Ableitung der Michaelis-Gleichung. Die der Ableitung der Michaelis-Gleichung zugrunde gelegten Voraussetzungen treffen kurz vor dem vollständigen Verbrauch eines Leiterpeptids in einer Reaktionsfolge und zu Beginn einer folgenden Abbaureaktion nicht zu. Dennoch lassen sich die Konzentrationsverläufe der Leiterpeptide während der Sequenzierung, auf der Basis der kinetischen Vorgaben die bei der Ableitung der Michaelis-Gleichung gemacht wurden, grob skizzieren. Abbildung 15 zeigt die ungefähre Konzentrationsentwicklung einiger Leiterpeptide bei einer Sequenzierung unter der vereinfachten Annahme, dass die Geschwindigkeitskonstanten aller Abspaltungsschritte in etwa gleich sind. Zur besseren Übersicht wurden die Konzentrationskurven für das Enzym und den Enzymsubstratkomplexe weggelassen. Aus Gleichung (7), sowie Abbildung 13 und Abbildung 14 lässt sich entnehmen, wie auch durch die Variation von Enzymkonzentration, beziehungsweise der Spaltungszeit Einfluss auf die Sequenzierung genommen werden kann. So bieten sich zwei Möglichkeiten, durch Rekonstruktion aus den Teilsequenzen eines Peptids der Gesamtsequenz zu ermitteln. Das Prinzip einer Spaltungskontrolle über die Reaktionszeit ist ebenfalls in Abbildung 15 dargestellt.

Abbildung 15: Prinzip der zeitabhängigen Reaktionskontrolle beim enzymatischen Leitersequenzieren

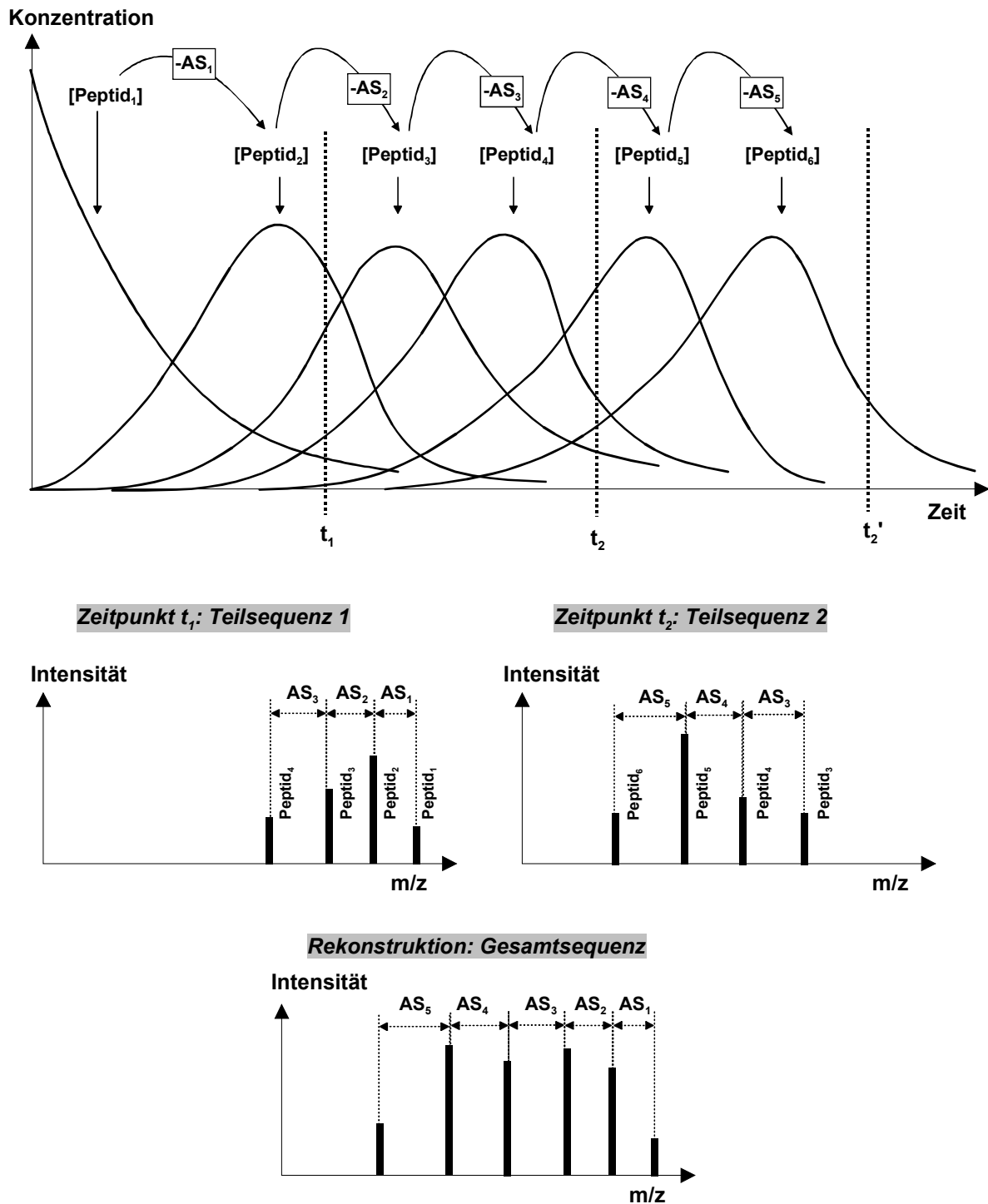
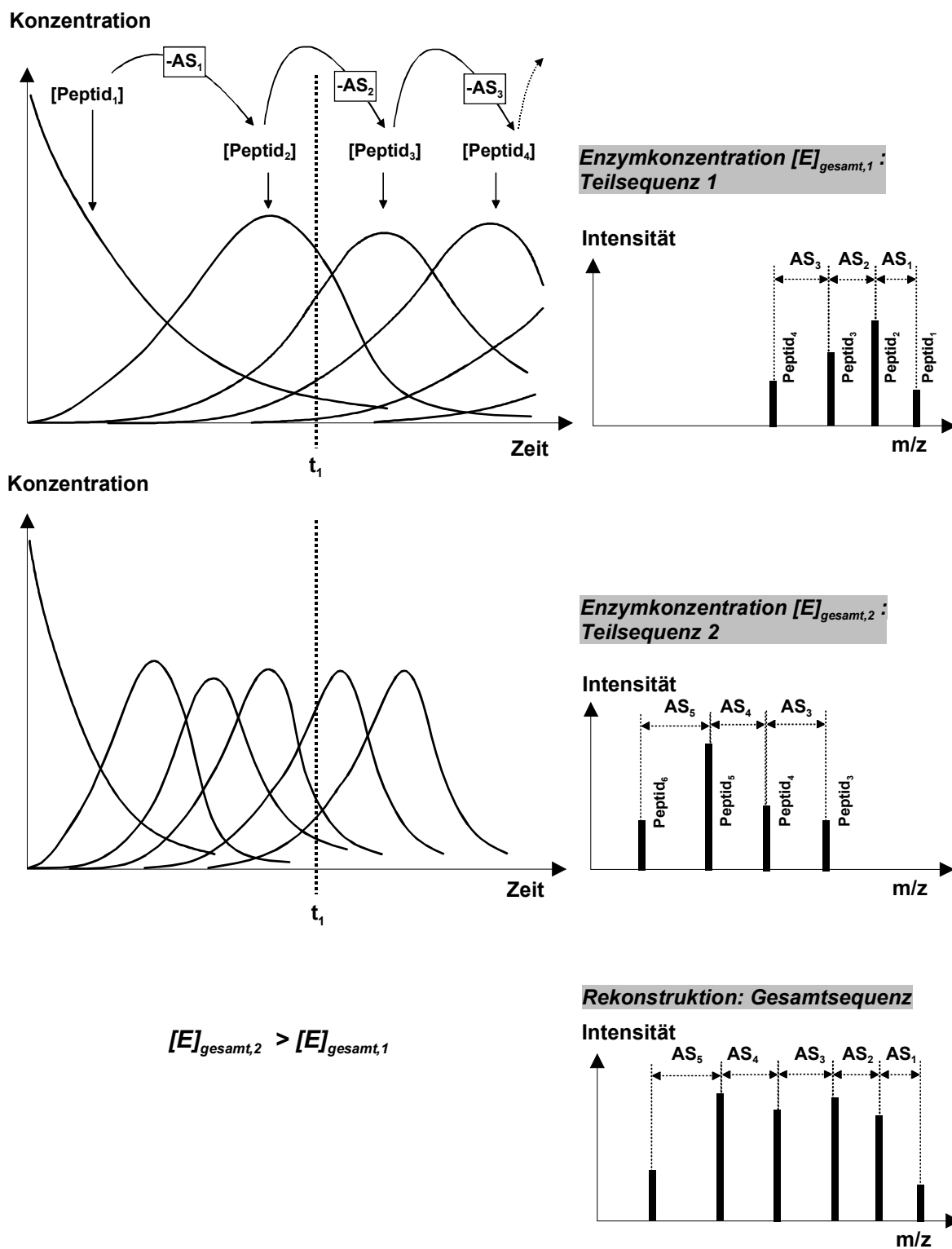


Abbildung 16 zeigt das Prinzip der konzentrationsabhängigen Steuerung. In diesem Fall ergeben sich die unterschiedlichen Teilsequenzen durch Anwendung verschiedener Enzymkonzentration bei gleicher Reaktionszeit.

Abbildung 16: Prinzip der konzentrationsabhängigen Reaktionskontrolle beim enzymatischen Leitersequenzieren



Eine höhere Enzymkonzentration hat nach der Michaelis-Gleichung ein größeres V_{max} und somit auch eine höhere Reaktionsgeschwindigkeit zur Folge. In Abbildung 14 entspricht eine Steigerung der Enzymkonzentration einem steileren Anstieg der Produkt- und einem steileren Abfall der Eduktkurve. Im komplexeren Fall der Leitersequenzierung ergibt sich dann, im Vergleich zu Abbildung 15, für zunehmend höhere Enzymkonzentration ein gestauchtes Bild der Konzentrationsverläufe.

Die Verhältnisse der zum jeweiligen Zeitpunkt t_1 oder t_2 bei zeitabhängiger Kontrolle bzw. $[E]_{gesamt,1}$ und $[E]_{gesamt,2}$ vorhandenen Konzentrationen der einzelnen Leiterpeptide sind in den Massenspektren durch unterschiedliche Peakhöhen wiedergegeben. Im Hinblick auf eine Massenanalyse mit MALDI jedoch ist dies nicht immer korrekt. Die MALDI-MS erlaubt in der Regel keine einfache Quantifizierung und die im Peakhöhen im Spektrum spiegeln daher nicht unbedingt die quantitativen Verhältnisse der Leiterpeptide wider.

Durch das unterschiedliche Abspaltungsverhalten der verschiedenen Aminosäuren je nach verwendetem Enzym sehen die in Abbildung 15 und Abbildung 16 dargestellten Kurven für den Realfall wesentlich unregelmäßiger und komplizierter aus. Einzelne intermediäre Peptidprodukte können bei sehr schneller Abspaltung der nächsten Aminosäure z.B. nur in sehr geringen Konzentrationen auftreten. In solch einem Fall können, wie in Abbildung 6 gezeigt, Sequenzlücken entstehen. Für das Experiment ist also die Wahl der richtigen Spaltungszeiten bei zeitabhängiger Kontrolle, oder richtig gestaffelter Enzymmengen für die Spaltung bei konzentrationsabhängiger Kontrolle, entscheidend für den Erfolg der Sequenzierung. Erfolgt beispielsweise die Bestimmung der zweiten Teilsequenz in Abbildung 15 statt bei t_2 zur Zeit t_2' , so wird auch hier eine Sequenzlücke entstehen, da die Zwischenprodukte Peptid₃, Peptid₄ und Peptid₅ bereits vollständig abgebaut wurden. Den gleichen Effekt hätte die Anwendung einer wesentlich größeren Enzymmenge $[E]_{gesamt,2}'$ statt $[E]_{gesamt,2}$ in Abbildung 16 – auch hier wären unter Umständen Peptide, die eine Überlappung der Teilsequenzen ermöglichen bei $[E]_{gesamt,2}'$ und t_1 bereits verschwunden.

Beide Arten der Reaktionssteuerung – zeitabhängig oder konzentrationsabhängig – sind sowohl für Sequenzierung in Lösung, als auch *on-target* geeignet. Bei der *on-target* Sequenzierung ist die Spaltungszeit zunächst definiert über die „natürliche“ Trocknungszeit des aufgetragenen Volumens der Enzymlösung unter den gegebenen Umgebungsbedingungen. Insbesondere Temperatur und Luftfeuchtigkeit sind hier ausschlaggebend. Ausgehend von diesem Zeitintervall kann die Reaktionszeit einerseits durch Zugabe von Säure oder anderen Inhibitoren verkürzt oder andererseits durch Zugabe von Wasser oder Puffer verlängert werden. Ebenso ist zu prüfen, über welchen Zeitraum das gewählte Enzym auf dem Proben-

träger überhaupt stabil, d.h. aktiv ist. Durch erneute Zugabe von Enzymlösung kann auch eine kombinierte zeit- und konzentrationsabhängige Spaltung erfolgen. Eine Steuerung der Kinetik über die Enzymkonzentration ist gerade bei der *on-target* Sequenzierung experimentell einfacher durchzuführen. Zudem spricht für eine Steuerung über die Enzymkonzentration auch noch der Zeitfaktor. Eine Beschleunigung der Reaktionsgeschwindigkeit mit höherer Enzymkonzentration liefert im konzentrationsabhängigen Experiment die gleiche Sequenzinformation viel schneller als in einem zeitabhängigen Experiment mit gleichbleibender Enzymkonzentration.

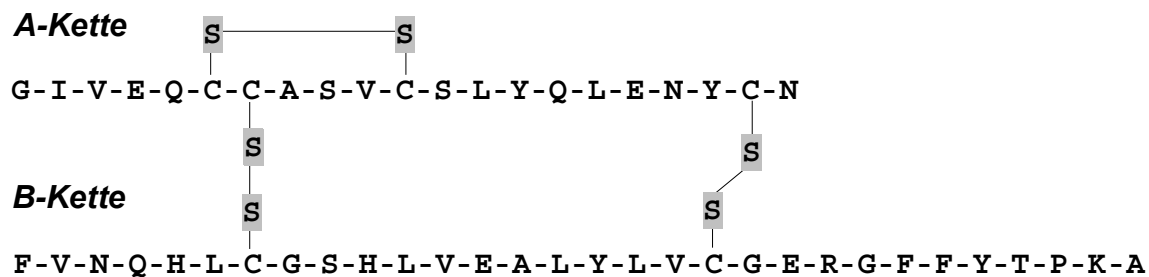
1.4 Weitere Aspekte der enzymatischen Leitersequenzierung

1.4.1 Proteinspaltung

1.4.1.1 Probleme bei der Sequenzierung intakter Proteine

Bereits die Betrachtung des analytisch nutzbaren Bereichs für die Leitersequenzierung unter Punkt 1.2.2.3 hat gezeigt, dass die enzymatische Sequenzierung intakter Proteinen mittels Leitersequenzierung aufgrund des begrenzten monoisotopischen Auflösungsvermögens der MALDI-TOF-MS nicht oder nur eingeschränkt möglich ist. Die Nutzung der Leitersequenzierung kann sich also nur auf sehr kleine Proteine im Bereich bis max. ca. 10 kDa mit einfacher Primärstruktur beschränken. Bei einer enzymatischen Sequenzierung intakter Proteine ergeben sich jedoch bereits vor der Massenanalyse Probleme, nämlich beim Abbauvorgang durch die Exopeptidasen. Sekundär- und / oder Tertiärstruktur behindern häufig die Zugänglichkeit der Proteintermini für die Exopeptidase. Denaturierende Detergenzien oder chaotrope Salze, die solche Strukturen auflösen können, führen jedoch selbst in geringen Konzentrationen zur Inaktivierung von Enzymen. Gerade Disulfidbrücken, die sekundäre Strukturmerkmale in wichtiger Weise mitbestimmen, bereiten bei der Sequenzierung Probleme, selbst wenn die Proteintermini zugänglich sind und so ein Abbau einzelner Aminosäuren aus den Protein direkt möglich ist. Spätestens am Cystin kommt bei allen Exopeptidasen der Abbau zum Stehen, da das so gebundene Cystein von keiner Exopeptidase abgebaut werden kann. Dieses Problem betrifft natürlich auch Disulfidbrücken, die unterschiedliche Proteinketten zusammenhalten, wie z.B. im Rinderinsulin oder β -Haptoglobin. Bei solchen Proteinen besteht zudem die Schwierigkeit, dass eine Sequenzierung an mehreren Termini gleichzeitig erfolgen kann und sich somit mehrere Leiterspektren überlagern. Bereits an einer so einfachen Proteinstruktur wie der des Rinderinsulins zeigen sich diese Probleme sehr deutlich:

Abbildung 17: Primärstruktur von Rinderinsulin



Symbole und Abkürzungen der Aminosäuren: siehe Anhang D

Bei der N-terminalen Sequenzierung lässt sich anhand der Komplexität des Leiterspektrums vermuten, dass es sich um Sequenzen aus mehreren Ketten handelt. Aber selbst wenn diese Information bekannt ist, wird es schwer sein, einzelne Aminosäuren aus dem Spektrum auszulesen oder gar einer bestimmten Kette zuzuordnen. Außerdem endet die Sequenz im obigen Beispiel spätestens nach dem Abbau von fünf Aminosäuren der A-Kette (GIVEQ) und sechs Aminosäuren der B-Kette (FVNQHL).

Aus den genannten Gründen kann die enzymatische Leitersequenzierung eines Proteins im Regelfall nur durch die Sequenzierung kleinerer, enzymatisch oder chemisch erzeugter Proteinfragmente erfolgreich verlaufen. Vorausgehend ist außerdem eine Denaturierung des Proteins sowie anschließende die Spaltung vorhandener Disulfidbrücken und Blockierung der erzeugten, freien Cystein-Thiolgruppen notwendig. Die Spaltung der Disulfidbrücken erfolgt entweder oxidativ (z.B. Perameisensäure), wobei hier auch gleichzeitig eine Blockierung der Cysteine stattfindet (Cysteinsäure), oder reduktiv, z.B. mit 1,4-Dithiothreitol, 2-Mercaptoethanol Tributylphosphin oder TCEP. Reduzierte Cysteine werden über Alkylierung, z.B. mit Vinylpyridin oder Iodacetamid blockiert. Für die Proteinspaltung stehen eine Vielzahl an Endopeptidasen und Reagenzien unterschiedlicher Spaltungsspezifität zur Verfügung.

Soll aus den Teilsequenzen der einzelnen Proteinfragmente die Protein-Gesamtsequenz rekonstruiert werden, so müssen aus zwei oder mehr Spaltungen des Proteins, mit Proteasen oder Reagenzien unterschiedlicher Spaltspezifität, überlappende Fragmente erzeugt werden.

1.4.1.2 Sequenzierung terminal modifizierter Proteine

Für eine reine Proteinidentifizierung über einen kurzen Sequenzabschnitt und anschließende Datenbankrecherche ist man auf die Sequenzinformation aus den terminalen Proteinregionen nicht zwingend angewiesen. Nach der Spaltung des betreffenden Proteins stehen im Normalfall mehrere Proteinfragmente für die Sequenzierung zur Verfügung. Die Sequenzinformationen, die diese Peptide liefern können, sind zum großen Teil redundant.

Möchte man jedoch bei einem unbekannten Protein eine Sequenzinformation über die N- oder C-terminale Region, so ist in diesem Fall ein nicht-modifizierter Terminus keine notwendige Voraussetzung für dessen Sequenzierung. Für die häufigsten Modifikationen, wie z.B. Acetylierung, Formylierung, N-terminalen Ringschluss zu pyro-Glutaminsäure oder C-terminale Amidierung stehen geeignete Exopeptidasen zur Verfügung, die diese Blockierung abbauen können. Aus Sicht der N-terminalen enzymatischen Sequenzierung benötigen zwar alle in Tabelle 3 aufgeführten Aminopeptidasen eine freie α -Aminogruppe für die Abspaltung, allerdings existieren geeignete Peptidasen, wie z.B. Acylaminosäure Peptidase (EC 3.4.19.1) oder Pyroglutamin-Aminopeptidase (EC 3.4.19.3), die speziell nur N-terminal acetylierte Aminosäuren oder Pyroglutaminsäure vom N-Terminus eines Peptids abspalten. Für die enzymatische Sequenzierung stellen solche Fälle lediglich eine leichte Anpassung der Methode dar, entweder durch Einbeziehen einer zusätzlichen Peptidase in die Enzymmischung (parallele Peptidasekombination) oder durch einen zusätzlich vorgeschalteten Schritt (sequenzielle Peptidasekombination). C-terminale Modifikationen wie z.B. amidierte Aminosäuren werden von vielen der in Tabelle 2 aufgelisteten Carboxypeptidasen abgebaut. Ein freier C-Terminus ist hier nicht immer eine notwendige Voraussetzung [Umetsu 1997]. Die Massendifferenz der Leiterpeptide bei Abspaltung der terminalen Aminosäure des Ausgangspeptids gibt eventuell auch direkt einen Hinweis auf die Art der Modifikation. So ist beispielsweise eine Massendifferenz von ca. 111,0 Da zwischen dem Ausgangspeptid und dem ersten Leiterpeptid bei N-terminaler Sequenzierung ein deutliches Zeichen für einen Pyroglutamatrest.

1.4.1.3 Wahl der Endopeptidase für die Proteinspaltung

Nach den unter 1.2.2.3 gemachten Aussagen spielt die Verteilung der auftretenden Peptidgrößen bei einer Proteinspaltung für den Erfolg der Sequenzierung eine wichtige Rolle. Während Leiterpeptide im Massenbereich unter 500 Da (Bereich ① in Abbildung 10) der Interpretation nicht zugänglich sind, kann für Peptide im Massenbereich zwischen ca. 500 und 1000 Da (Bereiche ② und ③) und über ca. 3500 – 4000 Da (Bereich ⑤) in unter Umständen zumindest teilweise eine verlässliche Sequenzinformation erhalten werden. Der „optimal-sequenzierbare Anteil“ einer Proteinspaltung ergibt sich nur aus denjenigen Peptiden, deren Masse im Bereich ④ liegt. Bei der Spaltung eines Proteins sollte die Spaltungsspezifität daher so gewählt werden, dass ein möglichst großer Anteil der Spaltfragmente mit seinen Massen in eben diesen Bereich liegt. Abbildung 18 und Tabelle 6 zeigen die Massenverteilung der Peptidfragmente bei Spaltung von 100 verschiedenen, zufällig ausgewählten Proteinen aus

unterschiedlichsten Organismen mit acht unterschiedlichen Endoproteinasen oder Spaltungsreagenzien.

Abbildung 18: Verteilung der Peptidfragmentgrößen bei unterschiedlichen Proteinspaltungen

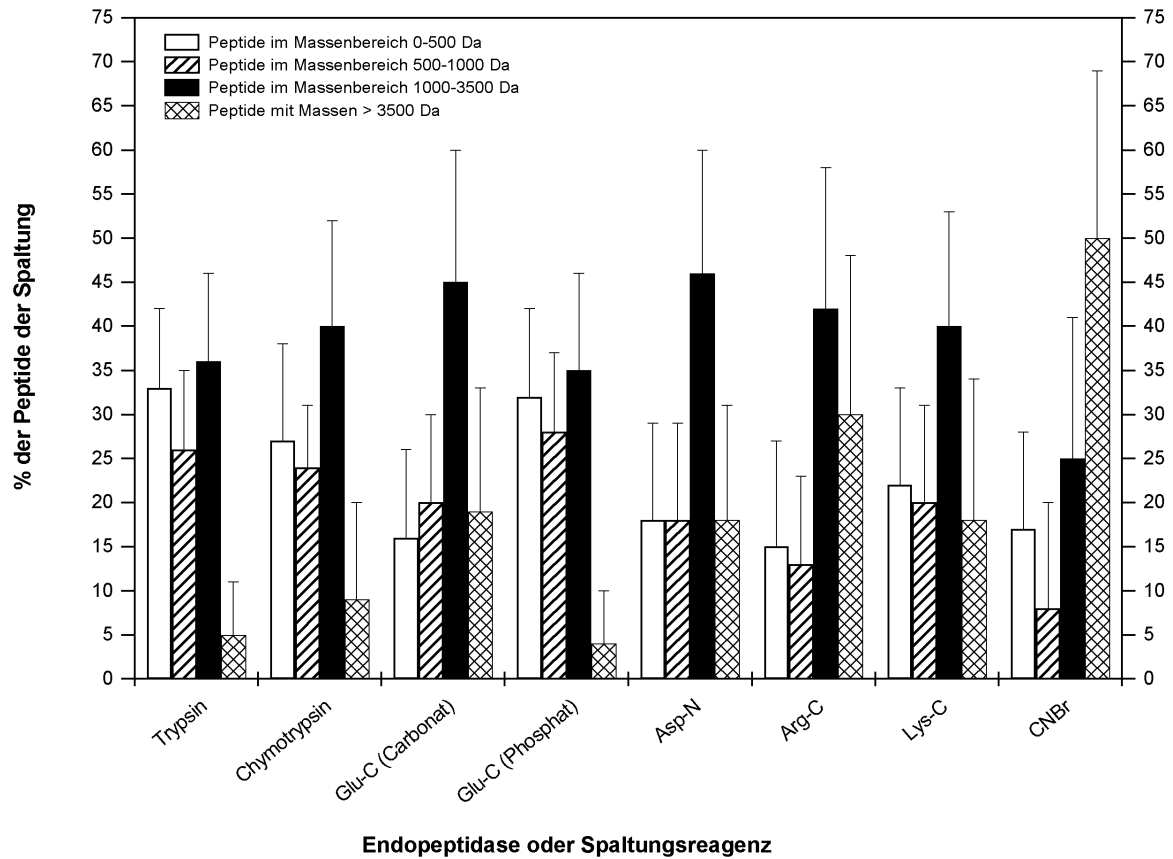


Tabelle 6: Verteilung der Peptidfragmentgrößen bei unterschiedlichen Proteinspaltungen

Enzym / Reagenz ¹	Spaltungsspezifität	Anteil der Peptide mit einer Masse im Bereich			
		0 – 500 Da	500 – 1000 Da	1000 – 3500 Da	> 3500 Da
Trypsin	nach Lys, Arg ²	33%	26%	36%	5%
Chymotrypsin	nach Phe, Tyr, Trp ²	27%	24%	40%	9%
GluC (Carbonat)	nach Glu ³	16%	20%	45%	19%
GluC (Phosphat)	nach Glu, Asp ³	32%	28%	35%	4%
AspN	vor Asp	18%	18%	46%	18%
ArgC	nach Arg ²	15%	13%	42%	30%
LysC	nach Lys ²	22%	20%	40%	18%
CNBr	nach Met	17%	8%	25%	50%

¹ Abkürzungen der Enzyme und Reagenzien siehe Anhang

² nicht vor Pro

³ nicht vor Glu oder Pro

Es wurden Proteine über einen großen Massenbereich von 7 bis 564 kDa ausgewertet. Die durchschnittliche Proteinmasse betrug ca. 57 kDa. Die Statistik wurde mit dem Programm GPMAW Version 4.11 durchgeführt. Unter Berücksichtigung der in Abbildung 18 eingezeichneten Standardabweichungen erscheinen lediglich drei der betrachteten Proteinspaltungen ungünstig, nämlich Spaltungen mit Trypsin, Endoproteinase GluC im Phosphatpuffer (GluC (Phosphat)) und Bromcyan. Bromcyan liefert einen viel zu hohen Anteil an sehr großen Fragmenten. Die genauere Aufschlüsselung für das Reagenz zeigt, dass je nach gespaltenem Protein häufig ein oder mehrere Fragmente entstehen, die sogar größer als 10000 Da sind.

Trypsin und Endoproteinase GluC (Phosphat) liefern auf der anderen Seite zwar die geringsten Anteile solcher großen Fragmente, jedoch liegt ca. ein Drittel aller proteolytischen Spaltpeptide in seiner Masse unter 500 Da und ist somit zur Gewinnung von Sequenzinformation unbrauchbar. Unter den proteolytischen Enzymen ist hier zugleich der Anteil der Peptide im optimalen Massenbereich am geringsten. Betrachtet man die Standardabweichungen (7-18% absolut) für alle proteolytischen Enzyme, so ist zwischen Chymotrypsin, Endoproteinase GluC im Carbonatpuffer (GluC (Carbonat)) und den Endoproteinasen AspN, ArgC und LysC kaum ein signifikanter Unterschied festzustellen. Lediglich bei Spaltung mit Chymotrypsin besteht der Nachteil, dass analog den Spaltungen mit Trypsin mit GluC (Phosphat) ein sehr hoher Anteil zu kleiner Fragmente entsteht. Chymotrypsin spaltet zudem auch häufig noch nach Leucin, was hier durch das Auswerteprogramm GPMAW nicht berücksichtigt wurde. Demzufolge dürften in der Praxis die Anteile für Spaltungen mit Chymotrypsin noch weiter zugunsten kleinerer Fragmente verschoben sein.

Aus Tabelle 6 lässt sich ein „sequenzierbarer Anteil“ für eine Endopeptidasespaltung abschätzen. Bei nachfolgenden Betrachtungen wird zunächst vereinfachend davon ausgegangen, dass alle Peptidfragmente der Proteinspaltung für die Sequenzierung zur Verfügung stehen und sich vollkommen durchsequenzieren lassen. Ein theoretisch maximal sequenzierbarer Anteil S_{max} errechnet sich dann wie folgt:

$$\begin{aligned}
 S_{max} = 100\% - & \quad (\text{Anteil der Peptide zwischen } 0 - 500 \text{ Da}) \\
 & - f_1 \cdot (\text{Anteil der Peptide zwischen } 500 - 1000 \text{ Da}) \\
 & - f_2 \cdot (\text{Anteil der Peptide } > 3500 \text{ Da})
 \end{aligned} \tag{8}$$

100% entspricht dabei der Summe aller Proteinfragmente

Der genaue Wert von f_1 hängt von der konkreten Massenverteilung der Peptidfragmente jedes einzelnen untersuchten Proteins in diesem Massenbereich ab. Wenn alle Fragmente zwischen ca. 500 und 1000 Da theoretisch um die maximale Masse 1000 Da liegen, so können die Peptidfragmente bis ca. 50% Sequenzinformation liefern. Unter alleiniger Berücksichtigung von f_1 bewegt sich S_{max} für proteolytische Spaltungen dann (nach Tabelle 6) zwischen ca. 50% für GluC (Phosphat) und 80% für ArgC. Für den Realfall wird man jedoch aus dem Peptidmassenbereich zwischen ca. 500 und 1000 Da wesentlich weniger Sequenzinformation erhalten. 25 – 40% ($f_1 \approx 0,25 - 0,4$) und somit ein S_{max} zwischen 45 und 75% stellen wesentlich realistischere Zahlen da. Der f_2 Faktor führt dazu, dass der Wert für S_{max} noch weiter nach unten korrigiert werden muss. So liefern zwar auch die Peptidfragmente über ca. 3500-4000 Da Sequenzinformation, allerdings wie unter Punkt 1.2.2.3 ausgeführt nur eingeschränkt. Auch hier entscheidet wiederum wie bereits für f_1 die individuelle Massenverteilung der Peptidfragmente jedes einzelnen Proteins in diesem Massenbereich. Je stärker die Verteilung zu größeren Massen verschoben ist, desto geringer wird der Wert von f_2 . Eine detaillierte Spaltanalyse der untersuchten 100 Proteine zeigt, dass bei Spaltungen mit ArgC und AspN die durchschnittliche Fragmentlänge im Massenbereich über 3500 Da größer ist als bei den restlichen Endopeptidasespaltungen. Ein konkreter Wertebereich für f_2 ist aufgrund des nach oben offenen Massenbereichs schwerer anzugeben als für f_1 , jedoch sollte $f_2 < 0,7 - 0,8$ nach einer Detailanalyse der betrachteten Proteinspaltungen in Abbildung 18 einem realistischen Bereich entsprechen.

Letztendlich lässt sich so die maximal erreichbare Sequenzabdeckung S_{max} bei enzymatischer Sequenzierung eines proteolytisch gespaltenen Proteins so auf ca. 40 – 70% abschätzen.

1.4.2 HPLC-Trennung der Proteinfragmente

1.4.2.1 Peaküberlappung der Proteinfragmente im Chromatogramm

Die erzeugten Proteinfragmente werden üblicherweise mittels C_{18} -Umkehrphasen (*reversed-phase*) Chromatographie getrennt. Die Chancen für eine erfolgreiche Sequenzierung werden sicherlich nur dann maximal sein, wenn die gesammelten Fraktionen jeweils nur eine Komponente enthalten. Fraktionen, die – bedingt durch partielle oder vollständige Peaküberlappung – zwei oder mehr Peptide gleichzeitig enthalten, können für die Interpretation der Leiterspektren problematisch sein, wenn sich auch die Sequenzen in ihren Leiterpeptidmassen überlappen.

Aus dem zugrundeliegenden Datenmaterial von Abbildung 18 kann die durchschnittliche Anzahl von Spaltpeptiden q bei einer Proteinspaltung in Abhängigkeit der Proteinmasse $M_{Protein}$ und der verwendeten Endopeptidase abgeschätzt werden (Formel 9). Die sich dabei ergebenden Werte für f sind in Tabelle 6 zusammengestellt.

$$q = \frac{M_{Protein}}{f} \quad (9)$$

q : Zahl der Fragmente, die bei der Proteinspaltung mit der betreffenden Endopeptidase entstehen
 $M_{Protein}$: Molare Masse des Proteins in kDa
 f : Endoproteinase-spezifischer Faktor

Tabelle 7: Faktoren zur Abschätzung der Peptidfragmentanzahlen bei unterschiedlichen Proteinspaltungen

Enzym / Reagenz	f	Standardabweichung
Trypsin	1,2	0,4
Chymotrypsin	1,4	0,7
Endoproteinase GluC (Carbonat)	2,3	1,2
Endoproteinase GluC (Phosphat)	1,1	0,3
Endoproteinase AspN	2,1	0,7
Endoproteinase ArgC	3,0	1,4
Endoproteinase LysC	2,2	1,3
CNBr	5,7	3,4

Das bedeutet beispielsweise, dass bei der Spaltung eines 50 kDa Proteins mit Trypsin im Mittel ca. 42 Peptidfragmente entstehen und über HPLC aufzutrennen sind, während es bei Endoproteinase LysC im Mittel nur ca. 23 sind. Die Möglichkeit einer vollständigen Auftrennung aller Peptide im Chromatogramm ist dabei im Zusammenhang mit der Peakkapazität und statistischen Auflösungswahrscheinlichkeit zu diskutieren. Die Peakkapazität n ist definiert über:

$$n = 1 + \frac{\sqrt{N}}{4} \cdot \ln(1 + k'_{max}) \quad (10)$$

N : Trennstufenzahl
 k'_{max} : maximaler k' -Wert

Nach Davis und Giddings [Davis 1983] lässt sich mit der Peakkapazität n auch die Wahrscheinlichkeit P berechnen, dass eine bestimmte Komponente eines Peptidgemischs als Einzelsubstanz eluiert und nicht von anderen Peptiden überlagert wird:

$$P \approx e^{-2q/n} \quad (11)$$

P : Wahrscheinlichkeit, dass ein einzelnes Peptid als Einzelsubstanz eluiert
 q : Anzahl der Peptide im Spaltgemisch

Die Wahrscheinlichkeit P' , dass alle Peptide im Gemisch getrennt werden, lässt sich nach Martin, Herman und Guichon [Martin 1986] über folgende Formel berechnen:

$$P' \approx \left(1 - \frac{q-1}{n-1}\right)^{q-2} \quad (12)$$

P' : Wahrscheinlichkeit, dass alle Peptide als Einzelsubstanzen eluieren

Wie sich diese Statistiken in der Praxis auswirken können, soll an einem HPLC-System mit typischen Kenngrößen für die bioanalytische Peptidtrennung aufgezeigt werden. Übliche Gradienten haben bei der Trennung von Peptidgemischen eine Dauer bis ca. 70min. Bei einer Totzeit von ca. 1-2min (je nach Systemkonfiguration) ergibt sich ein maximaler k' -Wert von ca. $k'_{max} \approx 70$. Die verwendeten C_{18} -reversed-phase Säulen mit 1-2mm Innendurchmesser und 100-150mm Säulenlänge haben Bodenzahlen N , die oft um 250000 liegen. Die aus diesen Werten zu errechnende Peakkapazität liegt bei ungefähr 450 zu. Mit diesen Werten lassen sich für Peptidgemische unterschiedlicher Komponentenzahl die Wahrscheinlichkeiten P und P' berechnen. In der Praxis wird man für die Sequenzierung am häufigsten Proteine mit einem Molekulargewicht bis 100 kDa zu tun haben. Dementsprechend sind nach Spaltung mit den in Tabelle 7 angegebenen Peptidasen maximal bis zu ca. 100 Spaltpeptide aufzutrennen. Abbildung 19 (folgende Seite) zeigt die resultierenden Auflösungswahrscheinlichkeiten P und P' für Gemische unter den oben angegebenen Bedingungen. Für Proteinspaltungen mit kleinerem f -Wert (Tabelle 7), in welchen mehr Fragmente für ein definiertes Protein entstehen, ist auch mit einer größeren Zahl an Peaks im Chromatogramm zu rechnen, die zwei oder mehr Komponenten enthalten. Für die Interpretation der möglicherweise überlappender Leitersequenzen kann dies Probleme zur Folge haben.

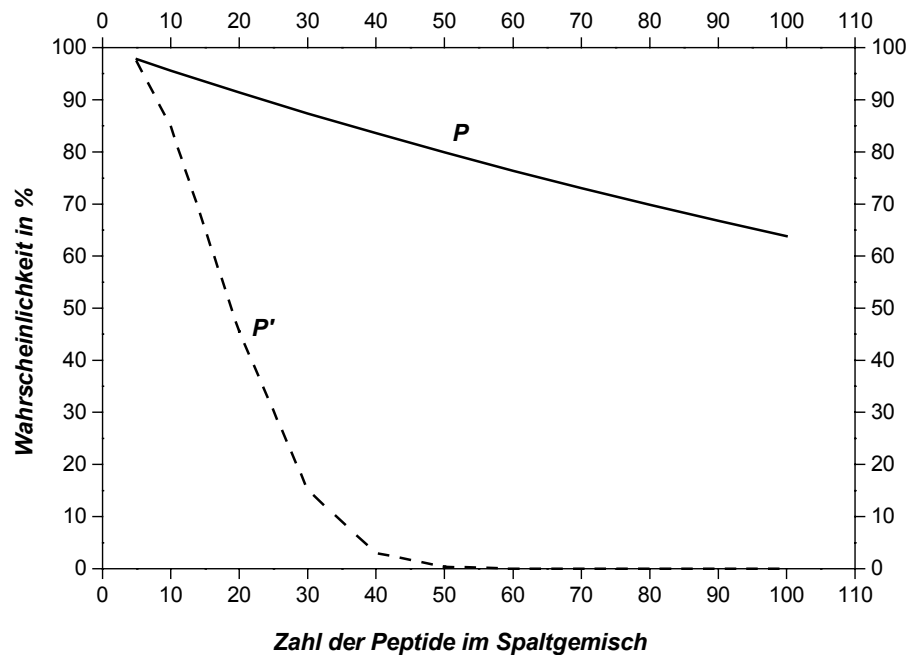
Abbildung 19: Peakauflösungswahrscheinlichkeiten P und P' für $k'_{\max} \approx 70$ und $N=250000$ 

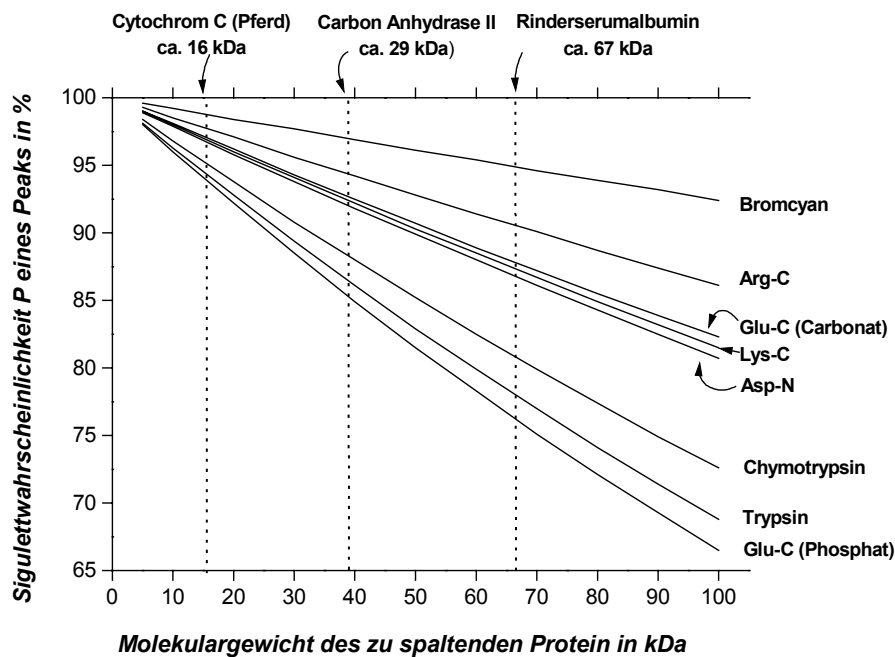
Abbildung 20 zeigt die Singulettwahrscheinlichkeiten, d.h. die Wahrscheinlichkeit, dass ein Peak im Chromatogramm auch nur ein Peptid enthält

- ❖ in Abhängigkeit der Masse des gespaltenen Proteins und
- ❖ in Abhängigkeit der angewendeten Spaltungsspezifität

berechnet aus den f -Werten von Tabelle 7 und P -Werten aus Abbildung 19.

Während es also z.B. für ein kleines Protein wie Cytochrom C möglich ist bei allen Spaltungen für über 93% der Proteinfragmente im Chromatogramm ein Singulett zu erhalten, sind es für ein großes Protein wie Rinderserum Albumin mit ca. 67 kDa nur noch 78%. Die restlichen 22% aller Peaks sind also ungenügend oder gar nicht getrennt, liefern also sogenannte Multipletts. Es ist fraglich, ob eine Sequenzinformation aus den Peptiden in diesen HPLC-Fractionen überhaupt erhalten werden kann. Aus Sicht einer sauberen HPLC-Trennung der Peptide wären also Spaltungen des Proteins vorteilhaft, die nur eine geringe Zahl an Spaltfragmenten liefern. Auch vorausschauend auf den Arbeits- und Zeitaufwand ist eine geringere Zahl zu sequenzierender Peptidfraktionen wünschenswert. Allerdings liefern Spaltungen mit solchen Eigenschaften (z.B. Endoproteinase ArgC oder CNBr) im Mittel wiederum größere Peptide, deren Leitersequenzierung im Zusammenhang mit MALDI-MS problematisch ist (geringere Auflösung der Peptidsignale; siehe oben).

Abbildung 20: Singulettwahrscheinlichkeiten eines HPLC-Peaks bei der Trennung von Peptiden aus unterschiedlichen Proteinspaltungen



1.4.2.2 Sequenzierbarer Anteil des Proteins

Unter dem Begriff des „sequenzierbaren Anteils“ soll hier derjenige Anteil des Proteins verstanden werden, der potentiell bei der Leitersequenzierung – auch bedingt durch die angewandten Methoden – Sequenzinformation liefern kann. So lassen sich sicherlich Tetra- oder Pentapeptide auch durch Exopeptidasen abbauen, Sequenzinformation erhält man jedoch aus der MALDI-MS, bedingt durch Interferenz der Peptidsignale mit Signalen der Matrixionen, nicht. Folglich gehören diese Peptide auch nicht zum sequenzierbaren Anteil. Der sequenzierbare Anteil der Gesamtproteins (entspricht 100%), verringert sich in der Praxis aus mehreren Gründen:

- (A) Peptidgrößenverteilung durch die angewandte Proteinspaltung
- (B) Multipletts bei der Chromatographie der Proteinfragmente
- (C) Wiederfindung der Proteinfragmente nach der Chromatographie

Punkt (A) wurde in vorletzten Abschnitt (1.4.1.3 Wahl der Endopeptidase für die Proteinspaltung) ausführlich diskutiert. Der durch die Wahl der Proteinspaltung resultierende, sequenzierbare Anteil wurde hier bereits mit S_{max} eingeführt und ließ sich auf höchstens 70% des Gesamtproteins abschätzen. Die Beträge, um die der sequenzierbare Anteil durch die

Einflüsse aus den Punkten (B) und (C) weiter reduziert wird, können nicht explizit angegeben werden – diese ergeben sich nur aus dem konkreten Experiment. Grund dafür ist, dass Beiträge aus (B) und (C) zum Teil bereits unter (A) berücksichtigt sind und sich zudem auch Beiträge aus (B) und (C) zum Teil überschneiden. So werden zum Beispiel viele kleine Peptide unter 500 Da im Startpeak des Chromatogramms koeluieren und erst gar nicht getrennt beziehungsweise gesammelt werden. Trotz solcher Überschneidungen ist anzunehmen, dass der sequenzierbare S_{max} Anteil durch reduzierende Beiträge aus (B) und (C) letztlich unter 70% liegt.

1.4.3 Verbesserungsmöglichkeiten der Methode durch Derivatisierungen

Viele Aminosäuren besitzen reaktive Gruppen, die in einfacher Weise eine chemische Modifizierung erlauben. Aus Sicht der Entwicklung einer Methode zur Leitersequenzierung gibt es mehrere Aspekte, die eine Derivatisierung mit geeigneten Reagenzien vorteilhaft erscheinen lassen.

- ❖ HPLC der Proteinfragmente
- ❖ Massenverschiebung der Leiterpeptide
- ❖ MS-Response der Leiterpeptide
- ❖ MS-Auswertung

Eine Derivatisierung des kompletten Spaltgemischs des Proteins vor der HPLC-Trennung der Spaltfragmente könnte beispielsweise eine empfindlichere Detektion erlauben. Möglich wäre dies durch Umsetzung mit Reagenzien, die chromophore oder fluorophore Gruppen mit charakteristischen Absorptions- bzw. Emissionsmaxima und hohen Extinktionskoeffizienten besitzen. Die empfindlichere Detektion erhöht unter Umständen auch die Wiederfindungsrate der Proteinfragmente nach der HPLC-Trennung und führt so zu einem insgesamt höheren, sequenzierbaren Anteil S_{max} .

Wesentlich entscheidendere Verbesserungen können sich durch eine Derivatisierung jedoch auf der eigentlichen Sequenzierungsseite der Methode im Zusammenhang mit verschiedenen Aspekten der massenspektrometrischen Detektion der Leiterpeptide ergeben. Betrachtet man die theoretischen Anteile kleiner Peptide bis 1000 Da die sich aus verschiedenen Möglichkeiten der Proteinspaltung ergeben (siehe Tabelle 6), so sind diese mit 25 – 60% extrem hoch. Reagenzien, die durch ihre Umsetzung mit dem Peptid eine nennenswerte Massenverschiebung zu höheren Massen zur Folge haben sind hier in zweierlei Hinsicht


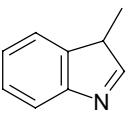
positiv zu bewerten. Einerseits können Fragmente unter 500 Da einer Sequenzierung zugänglich gemacht werden, andererseits kann die maximal lesbare Sequenzlänge der Fragmente zwischen 500 und 1000 Da vergrößert werden. Zusätzlich kann man je nach Derivatisierung auch mit einer Erhöhung oder Angleichung der MS-Response der einzelnen Leiterpeptide zu rechnen. Für solche Effekte sind mehrere Ursachen denkbar. Zum einen kann durch veränderte hydrophobe oder hydrophile Eigenschaften der modifizierten Peptide, je nach verwendeter MALDI-Matrix, ein besserer Einbau der Analytmoleküle in die Kristallstruktur der Matrix möglich sein (auch eine bessere Verteilung) und damit eine verbesserte Desorption [Cohen 1996]. Zum anderen kann eine verbesserte Ionisierung nach der Derivatisierung resultieren. Gerade für positive Ionen ist eine leichtere Bildung durch eingeführte Gruppen mit basischen Eigenschaften [Olumee 1995] oder delokalisierten Elektronensystemen plausibel. Einen ähnlichen Effekt könnte man auch beim Einbringen von Gruppen mit Atomen ausgedehnter Elektronenorbitale wie Brom oder Iod erwarten, denn auch diese Atome besitzen in der Gasphase eine gute Protonenaffinität [Huheey 1988]. Eine Derivatisierung mit Reagenzien, deren Struktur bereits eine positive Ladung aufweist und die diese Ladung auch ins derivatisierte Peptid miteinbringen, bezeichnet man in der Literatur als *charge-tagging* [Zenobi 1998]. Hier konnte bereits in einzelnen Fällen eine deutliche Verbesserung der MS-Response gezeigt werden [Liao 1995]. Ein solches *charge-tagging* im Zusammenhang mit der Leitersequenzierung kann weiterhin dazu dienen, die Intensitäten der Leiterpeptidsignale untereinander anzugleichen und somit den Einfluss der Aminosäuresequenz auf die Signalintensität zu vermindern. Damit kann beispielsweise die Abspaltung von Aminosäuren, die einen positiven Einfluss auf die Ionisierung besitzen (z.B. Lys oder Arg) im Verlauf der Sequenzierung kompensiert werden. Voraussetzung ist natürlich, dass die mit dem *charge-tag* modifizierte Aminosäure während der Sequenzierung nicht selbst abgespalten wird, sondern in allen Leiterpeptiden enthalten ist. Zudem kann ein *charge-tagging* aber auch die Leiterpeptidsignale im Vergleich zu Störsignalen von Puffer- und Matrixionen selektiv anheben und folglich die Unterdrückung der Leiterpeptidsignale minimieren oder vermeiden.

Ein weiterer Aspekt betrifft die vereinfachte Auswertung der Leiterspektren eines derivatisierten Peptids. Besitzt die eingeführte Gruppe ein Atom mit einer charakteristischen Isotopenverteilung, wie z.B. Chlor oder Brom, so ist es leichter, die entsprechenden Leiterpeptide im Spektrum von anderen, nicht-peptidischen Signalen zu unterscheiden. Eine besondere Hilfe bei der Spektreninterpretation sind auch Derivatisierung von Lysin, die nachfolgend eine simple Differenzierung vom sonst nahezu isobaren Glutamin erlauben (beide ca. 128,1 Da). Lysin besitzt im Gegensatz zu Glutamin eine sehr reaktionsfähige primäre Aminogruppe in der Seitenkette und lässt daher leicht und selektiv in Vergleich zu

Glutamin modifizieren [Thiede 1997, Bonetto 1997]. In einigen Fällen, wie z.B. beim Cystein, ermöglicht eine Derivatisierung angeblich überhaupt erst eine enzymatische Abspaltung der betroffenen Aminosäure [Bonetto 1997].

Als potentielle Derivatisierungsstellen eines Peptids oder Proteins stehen prinzipiell der freie N-Terminus oder reaktive Seitenketten der Aminosäuren zur Verfügung. Tabelle 8 zeigt eine Auswahl geeigneter, reaktiver Gruppen seitens der Aminosäuren und möglicher Derivatisierungsreagenzien.

Tabelle 8: Geeignete, reaktive Seitenketten der 20 natürlichen Aminosäuren für eine Derivatisierung

Aminosäure	reaktive Gruppe der Aminosäuren	mögliche Derivatisierungsreagenzien und deren reaktive Gruppen ¹
Serin, Threonin	-OH (aliphatisch)	<p><i>Säurehalogenide (z.B. Carbonsäure- oder Sulfonsäurechloride; X=Halogen)</i></p> $\text{R}-\text{C}(=\text{O})-\text{X} \quad \text{bzw.} \quad \text{R}-\text{SO}_2-\text{X}$
Tyrosin	-OH (aromatisch)	<p><i>Säureanhydride (z.B. Carbonsäureanhydride)</i></p> $\begin{array}{c} \text{R}_2-\text{C}(=\text{O})-\text{O} \\ \\ \text{R}_1-\text{C}(=\text{O})-\text{O} \end{array}$
Lysin	-NH ₂	<p><i>Isocyanate und Isothiocyanate</i></p> $\text{R}-\text{N}=\text{C}=\text{O} \quad \text{und} \quad \text{R}-\text{N}=\text{C}=\text{S}$
Methionin, Cystein	-S-CH ₃ / -SH	<p><i>Aldehyde</i></p> $\text{R}-\text{C}(=\text{O})-\text{H}$
Histidin	Imidazol 	
Tryptophan	Indol 	<p><i>N-Hydroxysuccinimidylcarbonate und -carbamate</i></p> $\text{R}-\text{O}-\text{C}(=\text{O})-\text{O}-\text{N} \begin{array}{c} \text{O} \\ \parallel \\ \text{C} \\ \parallel \\ \text{O} \end{array} \quad \text{und} \quad \text{R}-\text{N}-\text{C}(=\text{O})-\text{O}-\text{N} \begin{array}{c} \text{O} \\ \parallel \\ \text{C} \\ \parallel \\ \text{O} \end{array}$

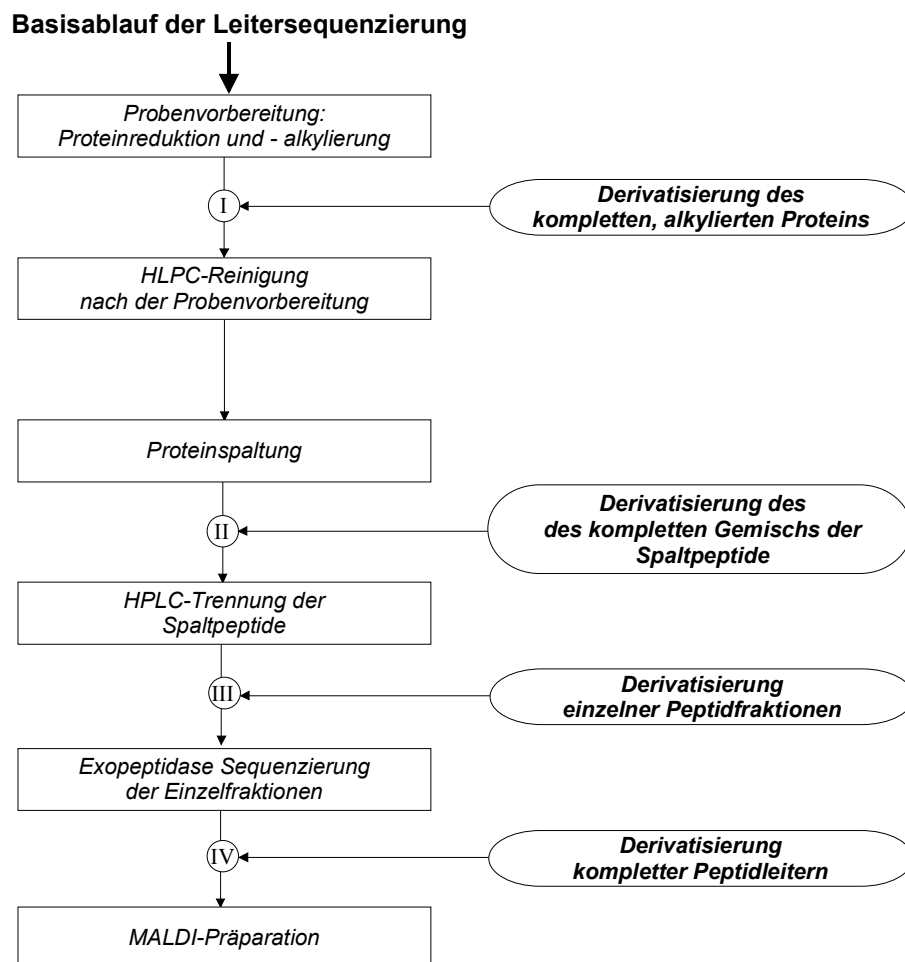
¹ R = organischer Rest des Moleküls

Unabhängig vom Typ der Reaktion und der an der Umsetzung beteiligten Gruppen, kommen nur solche Derivatisierungsreaktionen für eine Proteinsequenzierungsmethode in Frage, die folgende Forderungen weitgehend erfüllen:

- ❖ schnelle Reaktion
- ❖ schonende Umsetzung möglich, also z.B. keine Zerstörung von Peptidbindungen
- ❖ definierte Reaktion (eintretenden Veränderungen sind vorhersagbar und immer gleich)
- ❖ möglichst quantitative Umsetzung
- ❖ wenig oder keine Nebenprodukte (insbesondere keine störenden Nebenprodukte)
- ❖ keine Protease-Inhibition

Abbildung 21 zeigt, an welchen Stellen der bisher skizzierten Probenbearbeitung eine Derivatisierung für die Proteinsequenzierung prinzipiell möglich ist.

Abbildung 21: Möglichkeiten einer Derivatisierung im Ablauf einer Methode zur Leitersequenzierung



Bei einer Derivatisierung an den Punkten (I) und (III), also direkt vor einer proteolytischen Spaltung, muss damit rechnen werden, dass überschüssiges Reagenz, welches zumeist für eine quantitative Reaktion notwendig ist, die Protease inhibiert oder gar inaktiviert.

Bei (I) kann die Derivatisierung jedoch im Rahmen der Probenvorbereitung erfolgen. Hier findet anschließend ohnehin eine Reinigung des alkylierten Proteins statt. Eine Derivatisierung des gesamten Proteins am Punkt (I) ist jedoch hinsichtlich der nachfolgenden Endopeptidasespaltung nur eingeschränkt möglich. Eine Modifizierung reaktiver Seitenketten bestimmter Aminosäuren kann das Spaltmuster entscheidend verändern. So ist beispielsweise bei der Derivatisierung der ϵ -Aminogruppe am Lysin mit einem veränderten Spaltungsverhalten von Trypsin und Endoproteinase LysC zu rechnen, wenn eine Erkennung der modifizierten Aminosäure nicht mehr möglich ist. Die Proteinspaltung schlägt dann im Falle von Endoproteinase LysC entweder fehl oder findet, wie bei Trypsin, nur noch nach Arginin statt (entspricht einer Spezifität der Endoproteinase ArgC).

Erfolgt die Modifizierung am Punkt (III), so ist wahrscheinlich ein zusätzlicher Reinigungsschritt vor der Sequenzierung erforderlich. Eine Derivatisierung nach der Proteinspaltung bei (III), aber auch bei (II), führt zudem dazu, dass neben den reaktiven Seitenketten auch die freien N-Termini der Spaltpeptide umgesetzt werden. Für die N-terminale enzymatische Sequenzierung stellt dies zumeist (Ausnahme z.B. Acetylierung – siehe oben) eine irreversible Blockierung dar und die Sequenzierung ist dann nur noch C-terminal möglich.

Eine terminale Derivatisierung der Peptide ist aber jedoch zwingend notwendig, wenn man durch eine eingeführte Gruppe eine Massenverschiebung erreichen möchte, die alle Leiterpeptide umfasst. An jeder anderen Stelle im Peptid kann die eingeführte Gruppe, und damit die Massenverschiebung, durch den Abbau der modifizierten Aminosäure während der Sequenzierung verloren gehen. Terminale Peptidderivatisierungen können nach der Proteinspaltung an den Stellen (II) (III) und (IV) der Probenaufarbeitung erfolgen. Dabei erlaubt nur Variante (IV) eine Sequenzierung beider Peptidtermini der Spaltpeptide bei gleichzeitiger Einführung einer positiven Massenverschiebung der Leiterpeptide. Bei allen Derivatisierungen einer Aminosäure vor der Exopeptidasespaltung besteht zudem die Gefahr, dass die Exopeptidase die modifizierte Aminosäure nicht mehr abbauen kann. Auch hier ist Derivatisierung nach Variante (IV) die einzige Möglichkeit dem Problem aus dem Weg zu gehen.

2 Experimenteller Teil

2.1 Verwendete Proteine, Peptide, Enzyme und Chemikalien

2.1.1 Exopeptidasen

Die Bewertung der enzymatischen Aktivität über einen Standardtest kann für die Leitersequenzierung nur als Anhaltspunkt dienen, da sich die Definition der jeweiligen Aktivitätseinheit (Unit) nur auf ein bestimmtes Substrat unter genau definierten Umgebungsbedingungen (pH, Temperatur, Puffermilieu) bezieht. Ziel dieser Arbeit war es jedoch die Kinetik der Abspaltung soweit zu optimieren und kontrollieren, dass die Beobachtung der abgespalteten Aminosäuren ein möglichst vollständiges Bild der vorliegenden Aminosäuresequenz des sequenzierten Peptids ergibt. Die ausgearbeiteten Versuchsparameter sind daher von den Parametern der Aktivitätstests zunächst unabhängig.

Tabelle 9: Verwendete Aminopeptidasen

Name	Herkunft Organismus	Verwendete Abkürzung	EC- Nummer	angegebene, spezifische Aktivität	Bezugs- quelle
Leucin Aminopeptidase (Aminopeptidase M)	Schweinenieren Mikrosomen	APM	3.4.11.2	11 U/mg	¹
Leucin Aminopeptidase (Aminopeptidase M)	Schweinenieren Mikrosomen	APM	3.4.11.2	40 U/mg	²
Aminopeptidase	<i>aeromonas proteolytica</i>	AAP	3.4.11.10	117 U/mg	²
Aminopeptidase I	<i>streptomyces griseus</i>	API	keine EC Nummer	623 U/mg	²
Leucin Aminopeptidase	Schweinenieren Cytosol	LAP	3.4.11.1	320 U/mg	³
Leucin Aminopeptidase	Schweinenieren Cytosol	LAP	3.4.11.1	300 U/mg	²

¹ Roche, Penzberg (Boehringer, Mannheim)

² Sigma, Steinheim

³ Serva – Boehringer Ingelheim, Heidelberg

Tabelle 10: Unit Definitionen der Amino-peptidasen

Peptidase		Substrat	Produkte	pH	Temp.
APM ¹	Eine Unit katalysiert die Umsetzung von 1 µmol des angegebenen Substrats zu den angegebenen Produkten bei den angegebenen Werten von pH und Temperatur in einer Minute	Leucin-4-Nitroanilid	4-Nitroanilin L-Leucin	7,2	37°C
AAP		Leucin-4-Nitroanilid	4-Nitroanilin L-Leucin	8,0	25°C
API ²		Leucin-4-Nitroanilid	4-Nitroanilin L-Leucin	8,0	25°C
LAP		Leucinamid	Ammoniak L-Leucin	8,5	25°C

¹ Ca. 40% der angegebenen Aktivität bei 25°C² Ist spezifisch angegeben für 3mM Substratkonzentration**Tabelle 11: Verwendete Carboxypeptidasen**

Name	Herkunft Organismus	Verwendete Abkürzung	EC- Nummer	angegebene, spezifische Aktivität	Bezugs- quelle
Carboxypeptidase A	Rinderpankreas	CPA	3.4.17.1	35 U/mg	¹
Carboxypeptidase B	Schweinepankreas	CPB	3.4.17.2	150 U/mg	¹
Carboxypeptidase P (sequencing grade)	<i>penicillium</i> <i>janthinellum</i>	CPP	3.4.16.5	> 25 U/mg	¹
Carboxypeptidase W	Weizen	CPW	3.4.16.5	61 U/mg	²
Carboxypeptidase Y (sequencing grade)	Hefe	CPY	3.4.16.5	> 300 U/mg	¹
Carboxypeptidase Y	Bäckerhefe	CPY	3.4.16.5	100 U/mg	²

¹ Roche, Penzberg (Boehringer, Mannheim)² Sigma, Steinheim**Anmerkung:**

Das von Roche (bzw. Boehringer) unter dem Namen „Carboxypeptidase P“ vertriebene Enzym ist keine „Membran-Pro-X Carboxypeptidase“ wie dies aus einem Vergleich mit Tabelle 2 zu vermuten wäre. Boehringer gibt diesem Enzym die EC-Nummer 3.4.16.1 (jetzt 3.4.16.5 und 3.4.16.6), was auch mit der bei den Experimenten beobachteten Spaltungsspezifität übereinstimmt.

Tabelle 12: Unit Definitionen der Carboxypeptidasen

Peptidase		Substrat ¹	Produkte	pH	Temp.
CPA	Eine Unit katalysiert die Umsetzung von 1 µmol des angegebenen Substrats zu den angegebenen Produkten bei den angegebenen Werten von pH und Temperatur in einer Minute	Hippuryl-L-Phenylalanin	Hippursäure L-Phenylalanin	7,5	25°C
CPB		Hippuryl-L-Arginin	Hippursäure L-Phenylalanin	7,65	25°C
CPP		N-CBZ-Glu-Tyr	N-CBZ-L-Glutaminsäure L-Tyrosin	3,7	30°C
CPW		N-CBZ-Phe-Ala	N-CBZ-L-Phenylalanin L-Alanin	4,0	30°C
CPY		N-CBZ-Phe-Ala	N-CBZ-L-Phenylalanin L-Alanin	6,75	25°C

¹ CBZ = Carbobezoxy (Benzyloxycarbonyl)

2.1.2 Endopeptidasen

- Endoproteinase LysC (sequencing grade; *lysobacter enzymogenes*); EC 3.4.21.50
- Endoproteinase GluC (sequencing grade; *staphylococcus aureus* V8); EC 3.4.21.19
- Chymotrypsin (sequencing grade; Rinderpankreas); EC 3.4.21.1
- Trypsin (sequencing grade; Rinderpankreas); EC 3.4.21.4
- Endoproteinase AspN (sequencing grade; *pseudomonas fragi* Mutante); 3.4.24.33

Alle Endopeptidasen wurden von der Firma: Roche, Penzberg (ehemals Boehringer, Mannheim) bezogen.

2.1.3 Synthetische Peptide für die Sequenzierung

Tabelle 13 und Tabelle 14 geben einen Überblick über die sequenzierten Testpeptide, deren Namen und Bezugsquelle. Synthetische Peptide, die keinen Trivialnamen besitzen, wie z.B. Angiotensin I, erhielten, neben der fortlaufenden Nummer der Peptidsynthese des Peptid-Service am MPI für Biochemie, einen systematischen Namen, basierend auf der Zahl ihrer Aminosäuren, der gerundeten monoisotopischen Masse $[M+H]^+$ des einfach positiv geladenen Moleküls und eventueller Modifikationen. Das Nomenklatureschema ist nachfolgend dargestellt:

$M - PX - Y (Z)$

- P:** steht als Abkürzung für "Peptid"
M: Modifikationen (allgemein, keine Positionsangabe)
X: Zahl der Aminosäuren
Y: $[M+H]^+$ (monoisotopisch)
Z: NH_2 bei Carbonsäureamiden

Bei dieser Bezeichnungsweise ist die Position von M im Peptid bei modifizierten Peptiden nur bei N-terminalen Modifikationen in der üblichen Weise durch den Zusatz „N α “ gekennzeichnet. Eine komplette Abkürzungsliste der in dieser Arbeit vorkommenden Modifikationen findet sich in Anhang B. Sofern nicht speziell gekennzeichnet, besitzen die aufgelisteten Peptide am N-Terminus das freie Amin H_2N - und der C-Terminus die freie Carboxylgruppe $-COOH$. $-NH_2$ am C-Terminus kennzeichnet ein Carbonsäureamid.

Tabelle 13: Sequenzierte synthetische Peptide – nicht modifiziert

Bezeichnung	Sequenz	$[M+H]^+$ monoisotopisch in Da	Bezugs- quelle ¹
[Lys ⁶]-Angiotensin II, Fragment 1-6	DRVYIK	793,4572	PS576
[Tyr ¹ , Lys ⁶]-Angiotensin II, Fragment 1-6	YRVYIK	841,4936	PS577
[Leu ¹ , Lys ⁶]-Angiotensin II, Fragment 1-6	LRVYIK	791,5144	PS578
[Ala ¹ , Lys ⁶]-Angiotensin II, Fragment 1-6	ARVYIK	749,4674	PS579
[Gly ¹ , Lys ⁶]-Angiotensin II, Fragment 1-6	GRVYIK	735,4517	PS581
[His ¹ , Lys ⁶]-Angiotensin II, Fragment 1-6	HRVYIK	815,4892	PS580
[Arg ¹ , Lys ⁶]-Angiotensin II, Fragment 1-6	RRVYIK	834,5314	PS582
[Pro ¹ , Lys ⁶]-Angiotensin II, Fragment 1-6	PRVYIK	775,4831	PS583
[Thr ¹ , Lys ⁶]-Angiotensin II, Fragment 1-6	TRVYIK	779,4780	PS584
[Asn ¹ , Lys ⁶]-Angiotensin II, Fragment 1-6	NRVYIK	792,4732	PS585
Dynorphin A (Dyn-17)	YGGFLRRIRPKLKWDNQ	2147,1992	PS379
Dynorphin A Fragment 1-16 (Dyn-16)	GGFLRRIRPKLKWDNQ	1984,1359	PS378
Dynorphin A Fragment 1-15 (Dyn-15)	GIRFLRRIRPKLKWDNQ	1927,1144	PS377
Dynorphin A Fragment 1-14 (Dyn-14)	FLRRIRPKLKWDNQ	1870,0929	PS376
Dynorphin A Fragment 1-13 (Dyn-13)	LRRIRPKLKWDNQ	1723,0245	PS375
Dynorphin A Fragment 1-12 (Dyn-12)	RRIRPKLKWDNQ	1609,9404	PS374
Dynorphin A Fragment 1-11 (Dyn-11)	RIRPKLKWDNQ	1453,8393	PS373
Dynorphin A Fragment 1-10 (Dyn-10)	IRPKLKWDNQ	1297,7382	PS372
Dynorphin A Fragment 1-9 (Dyn-9)	RPKLKWDNQ	1184,6541	PS371
Dynorphin A Fragment 1-8 (Dyn-8)	PKLKWDNQ	1028,5530	PS370
Dynorphin A Fragment 1-7 (Dyn-7)	KLKWDNQ	931,5002	PS369
Dynorphin A Fragment 1-6 (Dyn-6)	LKWDNQ	803,4052	PS368
Insulin B-Kette, oxidiert	FVNQHLCGSHLVEALYLVCGERGFFYTPKA	3494,6513	²
Adrenocorticotrophic Hormon (ACTH) Fragment 18-39	RPVKVYPNGAEDESAEAFPLEF	2465,1989	²
Angiotensin 1	DRVYIHPFHL	1296,6853	²
Angiotensin 2	DRVYIHPF	1046,5423	²
Bradykinin	RPPGFSPFR	1060,5693	²
P2-322	RF	322,1879	²
P2-331	RR	331,2206	³
P3-487	RRR	487,3217	³
P4-487	RLKA	487,3356	PS613
P4-492	RLFG	492,2935	PS384

Bezeichnung	Sequenz	[M+H] ⁺ monoisotopisch in Da	Bezugs- quelle ¹
P4-496	RLHA	496,2995	PS614
P4-508	RLYG	508,2884	PS383
P4-545	RLWA	545,3199	PS616
P4-548	RFIL	548,3561	PS494
P4-643	RRRR	643,4228	³
P5-593	TRGIF	593,3412	PS478
P5-603	RESLV	603,3466	PS480
P5-634	RLFAK	634,4040	PS615
P5-800	RRRRR	799,5239	³
P6-841	YRVYIK	841,4946	PS577
P6-748	LSRFIL	748,4722	PS495
P8-997	KNYKESDI	996,5002	PS431
P9-1069	TKNYKQTSV	1068,5690	PS426
P10-1081	DFSALLSQIS	1080,5577	PS454
P10-1191	NKDKNQESDI	1191,5653	PS430
P10-1213	FLDTLVVLHR	1212,7105	PS563
P10-1219	NKKKKDETEV	1218,6695	PS429
P10-1259	NGFRRTNEHK	1258,6405	PS566
P10-1360	NKKKKKEYFF	1359,7790	PS432
P12-1303	AGHEYGAELER	1302,6078	PS187
P12-1395	REVHTNQDPLDA	1394,6664	PS320
P14-1624	ISSYQDAIEIEN	1623,7754	PS455
P14-1639	REDFSGLLPEEFIS	1638,8016	PS453
P14-1713	ESKFQQKLAEFTTR	1712,8973	PS452
P14-1720	QETFSDLWKLLPEN	1719,8595	PS451
P15-1812	IKVTLVFEHVDQDLR	1812,0020	PS323
P18-2107	KLMDLDVEQLGIPEQEYS	2107,0270	PS246
P24-2722	FEQRVNSDVLTVSTVNSQDQ VTQK	2722,3646	PS162
P24-2768	TVVDSGLRFRVSRQAFVKNIN LQYI	2767,5259	PS53
P24-2836	TIRNSGLRNIQPRFAKNPH LRYI	2835,5970	PS54
P31-3467	YSDMREANYIGSDKYFHARG NYDAAKRGPGG	3466,5909	PS252
P31-3649	YIFPVHWQFGQLDQHPIDGY LSHTELAPLRA	3648,8392	PS163
P37-4098	GYPPQQGYPPQQGYPPQQGY PPQQGYPPQQGYPPQQG	4097,8847	PS241

¹ „PS###“ bezeichnet die fortlaufende Synthesenummer bei Peptiden, die vom Peptidservice des MPI für Biochemie hergestellt wurden

² Sigma, Steinheim

³ Wissenschaftliche Arbeit aus der Abteilung von Prof. Dr. L. Moroder, MPI für Biochemie, Martinsried

Tabelle 14: Sequenzierte synthetische Peptide – modifiziert ¹

Bezeichnung	Sequenz	[M+H] ⁺ monoisotopisch in Da	Bezugs- quelle ²
Cys-Heat Shock Protein, 25kD, Fragment 81-93	CLNRQLS (phos) SGVSEIR	1641,7784	²
phos-P15-1474	TPSGQAGAAASES (phos) LF	1473,6263	PS847
phos-P13-1550	PHS (phos) HPALTPEQKK	1549,7528	PS846
phos-P10-1277	YESHGKLEY (phos) A	1276,5251	PS845
phos-P13-1560	LGEY (phos) GFQNALIVR	1559,7623	PS844
phos-P12-1440-NH ₂	PTSFGY (phos) DKPHVL-NH ₂	1439,6724	³
phos-P12-1445-NH ₂	VVHSGY (phos) RHQVPS-NH ₂	1444,6851	³
Nα-Ac-P7-808	Ac-GRDGFSSK	808,3953	PS850
Nα-Ac-P7-919	Ac-ALELFRK	918,5413	PS851
Nα-Ac-P12-1204	Ac-GASSGPTIEEVD	1203,5381	PS727
Nα-Ac-P14-1421	Ac-HGTVVLTALGGILK	1420,8528	PS849
Substance P	RPKPQQFFGLM-NH ₂	1347,7361	⁴
P12-1360-NH ₂	PTSFGYDKPHVL-NH ₂	1359,7001	³
P12-1365-NH ₂	VVHSGYRHQVPS-NH ₂	1364,7187	³
Neurotensin	(pyro) ELYENKPRRPYIL	1672,9176	⁴

¹ Modifikationen innerhalb der Sequenz stehen nach der betroffenen Aminosäure

² „PS###“ bezeichnet die fortlaufende Synthesenummer bei Peptiden, die vom Peptidservice des MPI für Biochemie, Martinsried hergestellt wurden

³ Wissenschaftliche Arbeit aus der Abteilung von Prof. Dr. L. Moroder, MPI für Biochemie, Martinsried

⁴ Sigma, Steinheim

2.1.4 Proteine für die Sequenzierung

- | | |
|--|---|
| • Myoglobin (Pferdeherz) | Sigma, Steinheim |
| • Cytochrom C (Pferdeherz) | Sigma, Steinheim |
| • α-Casein (Rindermilch) | Sigma, Steinheim |
| • β-Casein (Rindermilch) | Sigma, Steinheim |
| • β-Lactoglobulin (Rindermilch) | Sigma, Steinheim |
| • Alkoholdehydrogenase (Bäckerhefe) | Sigma, Steinheim |
| • Aldolase (Hasenmuskel) | Sigma, Steinheim |
| • Serumalbumin (Rind) | Gerbu Biotechnik, Heidelberg |
| • Serumalbumin (Schwein) | Sigma, Steinheim |
| • Serin Proteinase Inhibitor (<i>delonix regia</i> Samen)
„DrTI“ | zur Verfügung gestellt von
Maria Luiza V. Oliva ¹
Dep. Bioquímica Escola Paulista
de Médico, S. Paulo, SP. Brasil |

¹ Die in dieser Arbeit angegebene Aminosäuresequenz des DrTI wurde von Herrn Reinhard Mentele über Edman-Sequenzierung ermittelt und freundlicherweise zum Vergleich zur Verfügung gestellt.

2.1.5 Chemikalien

Lösungsmittel

- | | |
|--------------------------------|------------------|
| • Acetonitril (gradient grade) | Merck, Darmstadt |
| • Ethanol, absolut (p.a.) | Merck, Darmstadt |
| • 2-Propanol (p.a.) | Merck, Darmstadt |
| • Aceton (p.a.) | Merck, Darmstadt |

Für alle wässrigen Lösungen wurde deionisiertes Wasser aus einem Milli-Q Plus System (Millipore, Eschbronn), mit dem Aufarbeitungsmodul „Membra Pure“ (Membrane Pure, Bodenheim) verwendet.

Säuren, Basen und Puffersubstanzen

- | | |
|--|------------------------------|
| • Ameisensäure (p.a.) | Merck, Darmstadt |
| • Salzsäure (p.a.) | Riedel-de-Häen, Seelze |
| • Trifluoressigsäure (Protein Sequencer Grade) | Perkin Elmer, Warrington USA |
| • Natriumhydrogenphosphat Monohydrat (p.a.) | Merck, Darmstadt |
| • di-Natriumhydrogenphosphat (p.a.) | Merck, Darmstadt |
| • Ammoniumbicarbonat | Sigma, Steinheim |
| • Ammoniumacetat | Sigma, Steinheim |
| • Ammoniumchlorid | Sigma, Steinheim |
| • Ammoniumcitrat | Sigma, Steinheim |
| • Tris-(hydroxymethyl)-aminomethan (p.a.) | Riedel-de-Häen, Seelze |
| • Natriumhydrogencarbonat (p.a.) | Merck, Darmstadt |
| • Natriumhydroxid (p.a.) | Merck, Darmstadt |
| • N-Methylmorpholin | Merck, Darmstadt |
| • N-Methylpiperidin | Merck, Darmstadt |
| • Natriumborat | Waters, Milford, USA |

sonstige Chemikalien

- | | |
|--|------------------|
| • Ethylendiamintetraessigsäure-di-Natriumsalz (p.a.) | Merck, Darmstadt |
| • 2-Mercaptoethanol | Sigma, Steinheim |
| • 1,4-Dithioerythrit | Merck, Darmstadt |
| • Guanidiniumhydrochlorid | Merck, Darmstadt |
| • 4-Vinylpyridin | Sigma, Steinheim |
| • Magnesiumchlorid Hexahydrat (p.a.) | Merck, Darmstadt |
| • Mangan(II)-chlorid Dihydrat (p.a.) | Merck, Darmstadt |

Derivatisierungsreagenzien

- | | |
|---|----------------------|
| • Essigsäureanhydrid (p.a.) | Merck, Darmstadt |
| • Acetylchlorid (p.a.) | Merck, Darmstadt |
| • Dabsylchlorid | Sigma, Steinheim |
| • Dansylchlorid | Sigma, Steinheim |
| • 4-Brombenzoylchlorid | Merck, Darmstadt |
| • 3-Brombenzoylchlorid | Merck, Darmstadt |
| • 6-Bromhexanoylchlorid | Merck, Darmstadt |
| • Rhodamin B Isothiocyanat | Sigma, Steinheim |
| • Eosin-5-isothiocyanat | Sigma, Steinheim |
| • 2-Brombenzyloxycarbonyl-N-hydroxysuccinimid | Merck, Darmstadt |
| • 9-Fluorenmethylsuccinimidylcarbonat | Merck, Darmstadt |
| • 9-Fluorenmethylchloroformiat | Sigma, Steinheim |
| • AccQ-Fluor TM | Waters, Milford, USA |

MALDI-Matrices

- | | |
|--------------------------------------|-----------------------|
| • 2,5-Dihydroxybenzoesäure | Fluka, Buchs, Schweiz |
| • 2-Hydroxy-5-methoxybenzoesäure | Sigma, Steinheim |
| • α -Cyano-4-hydroxyzimtsäure | Sigma, Steinheim |
| • Bernsteinsäure (p.a.) | Merck, Darmstadt |

Gase

- | | |
|--------------|--------------------|
| • Helium 4.6 | Linde, Deutschland |
|--------------|--------------------|

2.2 Verwendete Geräte

2.2.1 HPLC

Für die Trennung der proteolytischen Fragmente:

Hewlett –Packard HP 1100 (Agilent Technologies, Palo Alto, Kalifornien, USA) – System 1

Hochdruckgradientensystem mit folgender Konfiguration:

- Autosampler
- Binäre Pumpe (Kolbenpumpe)
- Entgaser
- Diodenarray Detektor (DAD)
- HP ChemStation Software Rev. A. 06.03 [509] auf Windows NT 4.0

Pharmacia-LKB Smart System (Amersham Pharmacia, Uppsala, Schweden) – System 2

Hochdruckgradientensystem mit folgender Konfiguration:

- „µ Precision Pump“ – binäre Spritzenpumpe (2x10ml)
- „µ Separation Unit“ mit automatischem Fraktionssammler
- „µ Peak Monitor“ Detektor
- Smart Manager 1.51 Software auf OS/2

Verwendete Säulen für System 1 und System 2:

- | | |
|---|--------------------------|
| • C ₁₈ -Superspher 60 RP-select B 4µm, 100/2mm | Merck, Darmstadt |
| • C ₁₈ -Purospher endcapped Microcart [®] 150/1mm | Merck, Darmstadt |
| • C ₁₈ -AQ Reprosil-Pur 3µm, 150/1mm | Dr. A. Maisch, Ammerbuch |
| • C ₁₈ -AQ Reprosil-Pur 5µm, 125/1mm | Dr. A. Maisch, Ammerbuch |

Für die Proteinaufreinigung nach Reduktion und Alkylierung:

Pharmacia Biosystems (Amersham Pharmacia, Uppsala, Schweden) – System 3

Niederdruckgradientensystem mit folgender Konfiguration:

- Gradientenpumpe 2249, Pharmacia LKB
- Niederdruckmischer 11300 Ultrograd[®] Mixer, LKB Bromma
- Detektor VWM 2141, Pharmacia LKB
- Schreiber

Verwendete Säule für System 3:

- | | |
|--|-----------------------------------|
| • Aquapor [®] , RP-300, C8;
Porenweite 300Å, 7µm, 30/2,1mm | Perkin Elmer / Applied Biosystems |
|--|-----------------------------------|

2.2.2 Massenspektrometer

Als MALDI-Massenspektrometer wurde ein REFLEX III der Firma Bruker Daltonik (Bremen) eingesetzt. Es besitzt den in Abbildung 7 gezeigten Aufbau mit folgender Konfiguration:

- UV-Laser: Stickstofflaser „VSL 337ND“
Wellenlänge: 337nm
Pulsenergie: ca. 250µJ
Pulsdauer: 3ns FWHM
LSI Laser Science Incorporated, Newton, USA
- IR-Laser: Festkörperlaser Erbium-YAG „Speser Er:YAG Q“
Wellenlänge: 2,94µm
Pulsenergie: max. 15mJ
Pulsdauer: max. 80ns FWHM (Q-Switch)
Spektrum, Berlin
- Ionenquelle: SCOUT™ 26
- Probenträger (Targets): rund, mit Platz für 26 Proben
(26 vorgegebene Probenspots)
Material: rostfreier Edelstahl (stainless steel)
- TOF-Operationsmodi:
 - Linear-TOF
 - Reflektor-TOF (2-stufiges Reflektron)
- Detektoren: Dual Microchannel Plate
- Digitizer: 1GHz LeCroy 9350A
Chesnut Ridge, USA
- Analogsteuerung: 12 Channel High Resolution
Digital-to-Analog Converter (DAC)
- Beobachtungsoptik (Quelle): Color CCTV Camera „WV-CL352E“
Matsushita Electric, Osaka, Japan
- Bildschirm: PVM-1450QM Trinitron®, Color Video Monitor
Sony
- Primäre Akquisition: OS-9 (VME-Bus Computer) mit
OS-9 Reflex Data Acquisition Programm V. 20.16
- Akquisition und Auswertung: Sun SPARCstation 5:
UNIX System V Release 4.0 mit SunOS 5.6
SUN Acquisition : XACQ 4.0
SUN Auswertung: XMASS 5.0
- PC Auswertung: Bruker Data Analysis 1.6g

2.2.3 Sonstige Geräte und Hilfsmittel

Thermostatisierungen:	- Thermomixer 5437 Eppendorf, Hamburg
	- Thermoblock für MALDI-Targets und Reaktionsgefäße Werkstätten des MPI für Biochemie, Martinsried
Pipetten:	- P20, P200, P1000 Abimed, Langenfeld P2 und P10 Eppendorf, Hamburg
Pipettenspitzen:	0,1-10µl; 1-200µl; 100-1000µl Peske, Aindling
Präzisionsspritzen für die HPLC:	Unimetrics, Shorewood, USA <i>oder</i> Hamilton Bonaduz, Schweiz
Pasteurpipetten:	Volac – John Poulten LTD, Essex, Großbritannien
Reaktionsgefäße:	0,5ml; 1,5ml; 2,0ml Eppendorf, Hamburg <i>oder</i> Peske, Aindling
pH-Elektrode:	Kombination aus: Elektrode Blue Line 12pH Schott, Zwiesel Messgerät WTW pH521 Wissenschaftlich Technische Werkstätten,
Analysenwaagen:	- Sartorius Type 1419 Sartorius research Type R160P beide: Sartorius, Göttingen
Blot-Papier:	Immobilon-PSQ, PVDF-Transfermembran Porengröße: 0,45µm Millipore, Eschbronn
Leit-Tabs:	W. Plannet GmbH, Wetzlar
Filterpapier:	Schleicher & Schuell, Dassel
Spezielle Software:	GPMAW Version 4.12
Gefriertrocknung:	Lyovac GT 2E, Finn-Aqua
Heizschrank:	Heraeus

2.3 Vorbereitung der Proteinproben für die Sequenzierung

2.3.1 Reduktion und Alkylierung

Für die Reduktion mit 2-Mercaptoethanol wurde die eingewogene Menge des Proteins zunächst in 200µl einer Lösung aus 6M Guanidiniumhydrochlorid, 250mM Tris-HCl (pH 8,5) und 1mM EDTA aufgenommen. Die Reduktion erfolgte durch Zugabe von 5µl 2-Mercaptoethanol und anschließendes Erwärmen auf 50°C (2h) in Heizschrank. Bei der Reduktion mit DTT wurde das Protein in einer Lösung aus 6M Guanidiniumhydrochlorid, 250mM Tris-HCl (pH 8,5), 1mM EDTA und 10mM DTT aufgenommen und für 2h bei 60°C reduziert. Bei allen Versuchen betrug der molare Überschuss der Reduktionsmittels zum Protein mindestens 500.

Die reduzierten Proteinproben wurden anschließend auf Raumtemperatur abgekühlt und mit 5µl 4-Vinylpyridin versetzt. Die Alkylierungsreaktion wurde bei Raumtemperatur durchgeführt und nach 60min durch Zugabe von 20µl halb-konzentrierter, wässriger Ameisensäure (50%) abgestoppt. Der molare Überschuss an Vinylpyridin zum Protein betrug in allen Fällen mindestens 1000. Die reduzierten und alkylierten Proben wurden sofort anschließend auf dem HPLC-System 3 unter folgenden Bedingungen von überschüssigen Reagenzien gereinigt:

Säule:	Aquapor, 7µm Porenweite, 30/2,1mm
Gradient:	0%B $\xrightarrow{20\text{min}}$ 80%B
Eluent A:	Wasser + 0,08% TFA
Eluent B:	Acetonitril + 0,08% TFA
Fluss:	1ml/min
Detektionswellenlänge:	206nm

Alle gesammelten Proteinfraktionen wurden gefriergetrocknet und bis zu ihrer Verwendung bei -20°C aufbewahrt.

2.3.2 Endopeptidase Spaltungen

Endoproteinase LysC:

Endoproteinase LysC wurde in 50mM Tricin (pH 8,0) und 10mM EDTA aufgelöst und so eine Stammlösung mit einer Konzentration von 0,1µg/µl hergestellt. Das zu sequenzierende Protein wurde in 400µl eines wässrigen Verdauungspuffer, bestehend aus 25mM Tris-HCl (pH 8,5) und 1mM EDTA aufgenommen. Es wurde jeweils so viel Enzymlösung zur

Proteinlösung zugegeben, dass das Verhältnis von Enzymmenge zur Proteinmenge ca. 1:20 betrug. Bezogen wurde dabei auf die Proteineinwaage vor Reduktion und Alkylierung des Proteins. Die Inkubation der Mischung erfolgte anschließend bei 37°C für 18h, dann wurde die Spaltungsreaktion durch Zugabe von 30µl halb-konzentrierter, wässriger Ameisensäure (50%) abgestoppt.

Trypsin:

Trypsin wurde in 0,01% TFA aufgelöst und so eine Stammlösung der Konzentration 0,1µg/µl hergestellt. Das zu sequenzierende Protein wurde in 400µl eines wässrigen Verdauungspuffers, bestehend aus 100mM Tris-HCl (pH 8,05) gelöst. Es wurde jeweils so viel Enzymlösung zur Proteinlösung zugegeben, dass das Verhältnis von Enzymmenge zur Proteinmenge ca. 1:20 betrug. Bezogen wurde dabei auf die Proteineinwaage vor Reduktion und Alkylierung des Proteins. Die Inkubation erfolgte bei 37°C für 18h, dann wurde die Spaltungsreaktion durch Zugabe von 30µl halb-konzentrierter, wässriger Ameisensäure (50%) abgestoppt.

Chymotrypsin:

Chymotrypsin wurde in 50mM Tris-HCl (pH 8,0) aufgelöst und so eine Stammlösung mit einer Konzentration von 0,1µg/µl hergestellt. Das zu sequenzierende Protein wurde in 400µl eines wässrigen Verdauungspuffer, bestehend aus 50mM Tris-HCl (pH 8,0) aufgenommen. Es wurde jeweils so viel Enzymlösung zur Proteinlösung zugegeben, dass das Verhältnis von Enzymmenge zur Proteinmenge ca. 1:20 betrug. Bezogen wurde dabei auf die Proteineinwaage vor Reduktion und Alkylierung des Proteins. Die Inkubation erfolgte anschließend bei 37°C für 18h, dann wurde die Spaltungsreaktion durch Zugabe von 30µl halb-konzentrierter, wässriger Ameisensäure (50%) abgestoppt.

Endoproteinase GluC:

Spaltungen mit Endoproteinase GluC wurden im Phosphatpuffer ausgeführt. Dazu wurde Endoproteinase GluC in einem wässrigen Puffer aus 25mM Natriumphosphat ($\text{NaH}_2\text{PO}_4/\text{Na}_2\text{HPO}_4$) aufgelöst, so dass eine Stammlösung mit einer Konzentration von 1µg/µl resultierte. Der pH-Wert wurde auf 7,8 eingestellt. Im 400µl des gleichen Puffer wurde das zu sequenzierende Protein aufgenommen. Es wurde jeweils so viel Enzymlösung zur Proteinlösung zugegeben, dass das Verhältnis von Enzymmenge zur Proteinmenge mindestens 1:10 betrug. Bezogen wurde dabei auf die Proteineinwaage vor Reduktion und

Alkylierung des Proteins. Die Inkubation erfolgte anschließend bei 25°C für 18h, dann wurde die Spaltungsreaktion durch Zugabe von 30µl halb-konzentrierter, wässriger Ameisensäure (50%) abgestoppt.

Endoproteinase AspN:

Endoproteinase AspN wurde in 10mM Tris-HCl (pH 7,5) aufgelöst und so eine Stammlösung mit einer Konzentration von 0,04µg/µl hergestellt. Das zu sequenzierende Protein wurde in 400µl eines wässrigen Verdauungspuffer, bestehend aus 50mM Natriumphosphat (pH 8,0) aufgenommen. Es wurde jeweils so viel Enzymlösung zur Proteinlösung zugegeben, dass das Verhältnis von Enzymmenge zur Proteinmenge mindestens 1:10 betrug. Bezogen wurde dabei auf die Proteineinwaage vor Reduktion und Alkylierung des Proteins. Die Inkubation der Mischung erfolgte anschließend bei 37°C für 24h, dann wurde die Spaltungsreaktion durch Zugabe von 30µl halb-konzentrierter, wässriger Ameisensäure (50%) abgestoppt.

Alle Proteinspaltungen wurden bis zu ihrer Verwendung in Lösung bei –20°C aufbewahrt.

2.3.3 HPLC-Trennungen der proteolytischen Fragmente

Die HPLC-Trennungen der proteolytischen Spaltungen wurden auf System 1 oder System 2 (siehe Punkt 2.2.1) mit den folgenden Gradienten durchgeführt:

2mm Säule (nur auf System 2 angewendet):

0%B $\xrightarrow{80\text{min}}$ 60%B $\xrightarrow{5\text{min}}$ 70%B

Eluent A: Wasser + 0,08% TFA

Eluent B: Acetonitril + 0,08% TFA

Fluss: 200µl/min

Detektionswellenlängen: 206nm, 254nm, 280nm

1mm Säulen (auf System 1 und System 2 angewendet):

0%B $\xrightarrow{80\text{min}}$ 60%B $\xrightarrow{5\text{min}}$ 80%B $\xrightarrow{10\text{min}}$ 80%B

Eluent A: Wasser + 0,08% TFA

Eluent B: Acetonitril + 0,08% TFA

Fluss: 60µl/min

Detektionswellenlängen: 206nm, 254nm, 280nm (System 2: DAD 180nm-300nm)

Die eluierenden Peptidfraktionen wurden auf System 1 „von Hand“ gesammelt. Auf System 2 erfolgte die Fraktionierung und das Sammeln der Peaks automatisch über den Fraktionssammler in Kombination mit der Peakerkennung der SMART Software. Die gesammelten Fraktionen wurden bei -20°C gelagert.

2.4 MALDI-Probenpräparationen

2.4.1 Matrixlösungen

Alle Matrices wurden ohne weitere Aufreinigung verwendet. Im einzelnen wurden folgende Matrixlösungen für UV- oder IR-MALDI-MS Messungen auf dem Metalltarget hergestellt:

- ❖ CHCA-Matrixlösung für Peptidmessungen mittels UV-MALDI in einer Konzentration von 5mg/ml in Acetonitril/Wasser 50/50 (v/v) + 0,1% TFA;
- ❖ DHB-Matrixlösung für Protein- und Peptidmessungen mittels UV- und IR-MALDI in einer Konzentration von 20mg/ml in Acetonitril/Wasser 30/70 (v/v) + 0,1% TFA;
- ❖ DHBs-Matrixlösung wurde für Peptidmessungen mittels UV-MALDI verwendet. Für die Matrixlösung wurden zunächst eine DHB- und eine HMBS-Lösung mit Konzentration von jeweils 20mg/ml in Acetonitril/Wasser 30/70 (v/v) + 0,1% TFA hergestellt. Anschließend wurde die DHBs-Matrixlösung durch eine 9:1 Mischung der DHB- mit der HMBS-Lösung hergestellt.

Für die UV-MALDI-MS Messung auf der Membran wurde eine spezielle Matrixlösung mit CHCA hergestellt, welche CHCA in einer Konzentration von 20mg/ml in Methanol/Toluol 50/50 (v/v) + 0,1% TFA enthielt.

2.4.2 Standard Probenpräparationen

Die im folgenden beschriebenen Präparationen wurden, sofern nicht anders angegeben, für die UV- und IR-MALDI-MS Messung aller Proteine und Peptide, einschließlich der Leiterpeptide, angewendet.

Präparation in Lösung:

Die jeweilige Matrixlösung wurde mit der Lösung des Peptids im Volumenverhältnis 9:1 bis 1:1 in einem 0,5µl Reaktionsgefäß vorgemischt. Dazu wurde die Mischung nach dem Zusammengeben der beiden Lösungen im Reaktionsgefäß mehrmals mit einer Pipette aufgesaugt und wieder abgegeben. Die Präparation auf dem MALDI-Target erfolgte nach der

sog. „*dried-drop*“ Methode [Cohen 1996], indem 0,5-1,0µl der Mischung auf das MALDI-Target aufgetragen und einfach unter den gegebenen Umgebungsbedingungen (Raumtemperatur und Luftfeuchtigkeit) getrocknet wurden. Für die (Kontroll-)Messung von Proteinproben wurde ausschließlich diese Präparationsmethode in Lösung angewendet.

Präparation auf dem MALDI-Target (*on-target*)

Hier handelt es sich um eine Variante der „*dried-drop*“ Methode. 0,5-1,0µl der Probenlösung wurden auf das MALDI-Target aufgetragen und unter den gegebenen Umgebungsbedingungen getrocknet. Danach wurden 0,7-1,3µl der Matrixlösung auf den Probenspot pipettiert und wiederum bei Umgebungsbedingungen getrocknet. Nach der Auftagen der Matrixlösung auf das Target erfolgte keine vergleichbare Durchmischung mit der Pipette wie im Fall der Lösungspräparation.

Präparation auf der PVDF-Membran

Die Präparation wurde ausgehend von Beschreibungen in [Schreiner 1996] und [Vestling 1994] durch mehrere Versuch so abgeändert und optimiert, dass auch die Sequenzierungen von Proben auf der Membran gute Resultate bei der MALDI-MS Messung zeigten. Als erstes wurde auf einem mit Wasser befeuchteten Filterpapier (1,0cm x 1,0cm) ein Stück der PVDF Membran (Immobilon PSQ) der Größe 0,5cm x 0,5cm aufgelegt. Die gesamte Membran wurde mit 30µl Methanol (100%) benetzt. Noch kurz vor dem vollständigen Abtrocknen des Methanols wurde 1µl der Probenlösung auf die Membran gespottet. Nach dem Eintrocknen der Probe wurde 2 x 1µl Matrixlösung auf den Probenspot aufgetragen.

2.4.3 Präparationen proteolytischer Fragmente der Endopeptidasespaltungen

Proteolytische Fragmente der Endopeptidasespaltungen der untersuchten Proteine wurden vor (*mass-fingerprint* des Gemischs) und nach der *reversed-phase* HPLC-Trennung mittels MALDI-MS untersucht. Proben der kompletten Spaltmischungen sowie der Peptidfraktionen wurden mit CHCA- und / oder DHB-Matrix gemessen. Die Präparation erfolgte hier immer analog zu Punkt 2.4.2 (Präparation auf dem MALDI-Target): 0,2 – 1,0µl des Peptidgemischs oder einer Peptidfraktion wurden auf das MALDI-Target aufgetragen und getrocknet. Danach wurde 1µl der betreffenden Matrixlösung auf den Spot aufgetragen und ebenfalls getrocknet. Bei zu hoher Konzentrationen wurden verschiedene Verdünnungen (1:10 bis 1:100) des Spaltgemischs oder der Fraktion mit Acetonitril/Wasser 50/50 (v/v) + 0,1% TFA gemessen.

2.5 Enzymatische Leitersequenzierung

2.5.1 Präparationen für die Sequenzierung

2.5.1.1 Sequenzierung in Lösung

Die Sequenzierung in Lösung erfolgte in Analogie zum gezeigten Schema in Abbildung 11 (S. 41). Zunächst erfolgt eine Umpufferung des Peptids, in ein für die Sequenzierung kompatibels Puffersystem. Dazu wurde ein Aliquot von 1,0 – 10µl der Peptidlösung in einem offenen 0,5ml Eppendorfgefäß durch Abdampfen des Lösungsmittels bei 35 – 40°C bis zur Trockene eingengt und anschließend in einem rein wässrigen Puffersystem für die Exopeptidasespaltung wieder aufgenommen. Eine Einengung durch Vakuumzentrifugation (*speed-vac*) erweist sich im Vergleich zum einfachen Verdampfen als nachteilig. Die Signalintensitäten einer nachfolgenden MALDI-MS Messung zeigten sich im Mittel nach einer Einengung der Lösung über Vakuumzentrifugation immer deutlich geringer. Anscheinend sind bei der *speed-vac* Behandlung Substanzverluste durch Adsorption an den Gefäßwänden größer. Nach kurzem, intensiven Mischen der umgepufferten Probenlösung in einem Vortex-Mixer (ca. 10sec auf höchster Stufe) wurde ca. 1 – 2min bei der Temperatur für die Exopeptidasespaltung vortemperiert. Nun wurde soviel Volumen der möglichst konzentrierten Enzymlösung zugegeben, dass sich das gewünschte Enzym-Substrat-Verhältnis einstellte. Die Spaltung erfolgt im geschlossenen Reaktionsgefäß, welches in einen Thermoblock eingestellt wurde. Durch Zugabe verdünnter, wässriger TFA Lösung (0,1 – 1%, pH-Wert Absenkung auf pH 2 – 3) wurde die Sequenzierung abgestoppt. Die Spaltungstemperatur lag in allen Fällen im Bereich zwischen 25 – 45°C. 0,5 – 1,0µl des Spaltansatzes wurden dann, wie unter 2.4.2 (siehe oben) entweder in Lösung oder direkt auf dem Target präpariert.

Die bei der Sequenzierung verwendeten Volumina und Konzentrationen des wässrigen Puffersystems und der Enzymlösung, sowie die Inkubationstemperatur der Spaltung hängen vom jeweils verwendeten Enzym bzw. auch vom gewünschten Ausmaß der Sequenzierung ab. Eine Angabe der getesteten Versuchsparameter für die Sequenzierungen mit den unterschiedlichen Peptidasen folgt unten (Punkt 2.5.2, S. 84)

2.5.1.2 Sequenzierung auf dem MALDI-Target (*on-target*)

Die *on-target* Sequenzierung wurde ebenfalls in Anlehnung an das Schema in Abbildung 11 (S. 41) durchgeführt. Das MALDI-Target wurde in die Halterung eines Thermoblocks gelegt und auf die Spaltungstemperatur vorgewärmt. Die Spaltungstemperatur lag in allen Fällen im Bereich zwischen 25 – 45°C. 0,2 – 1,0µl der Peptidlösung wurden dann auf das MALDI-Target pipettiert und solange gewartet, bis das Lösungsmittel vollständig verdampft war. Dann wurden 0,8 – 1,2µl der betreffenden Enzymlösung im Spaltpuffer auf den getrockneten Spot aufgetragen. In den meisten Fällen wurde ein Volumen von 1,0µl verwendet. Im allgemeinen richtete sich die Reaktionszeit bei dieser Art der Sequenzierung zunächst nach der Zeit, die bis zum vollständigen Eintrocknen der Enzymlösung verstrich. Dieses Zeitintervall hing im wesentlichen vom aufgetragenen Volumen der Enzymlösung und der Spaltungstemperatur ab. Eine Verlängerung der Reaktionszeit kann durch eine oder mehrere der nachfolgenden Techniken erfolgen:

- ❖ Erneute Zugabe von Wasser
- ❖ Erneute Zugabe von Spaltpuffer
- ❖ Erneute Zugabe von Enzymlösung

Für die erneute Zugabe einer Lösung wurde ebenfalls ein Volumen zwischen 0,8 und 1,2µl gewählt, standardmäßig 1,0µl. Ein vorzeitiges Abstoppen einer Reaktion erfolgte direkt durch Zugabe von 0,7 – 1,3µl der Matrixlösung zum Spaltansatz auf dem MALDI-Target. Auf diese Weise erfolgt gleichzeitig die Präparation für die nachfolgende MALDI-Messung. Die Zugabe der Matrix zur sequenzierten Probe erfolgt bei Raumtemperatur (ca. 25 – 28°C) sonst erst nach dem Eintrocknen der Enzymlösung analog der Beschreibung unter 2.4.2 (Präparation auf dem MALDI-Target).

2.5.1.3 Sequenzierung auf der PVDF-Membranen

Die Präparation für die Sequenzierung auf der PVDF-Membran erfolgte in Anlehnung an das in Abbildung 12 (S. 44) gezeigte Schema, sowie die Beschreibung der „Präparation auf der PVDF-Membran“ unter Punkt 2.4.2 (siehe oben). Die Sequenzierungen auf der Membran wurden ausschließlich bei Raumtemperatur (25°C) durchgeführt. Nach dem Auftragen der Probe auf die Membran und deren Eintrocknen erfolgte für die Sequenzierung zusätzlich ein weiterer Benetzungsschritt. Dafür wurde auf die Probe zunächst 1,0µl Methanol (100%) und unmittelbar nach dem (sichtbaren) Abtrocknen des Methanols die Exopeptidase aufgetragen.

Der getrocknete Membranstreifen wurde dreimal mit je 100µl Wasser gewaschen, abschließend wurde 2 x 1µl Matrixlösung auf den Probenspot aufgetragen.

2.5.2 Anwendungsformen der Exopeptidasen – Einzelenzyme und Enzymkombinationen

Die folgenden vier Tabellen enthalten Informationen über Testbedingungen der verwendeten Exopeptidasen. Ähnliche Experimente wurden hierbei zu Gruppen zusammengefasst. Unterschieden wurde dabei neben N- und C-terminaler Sequenzierung außerdem nach der Anwendung von Einzelenzyme und Enzymkombinationen.

N-terminale Sequenzierungen:

Tabelle 15: Sequenzierungsversuche mit einzelnen Amino-peptidasen

Gruppen Bezeichnung	Experiment Bezeichnung	angewendeter Aktivitätsbereich [mU]	Puffersalz- konzentration [mM]	pH Bereich
APM	APM ₁	0,02 – 4,0	Tris : 10 – 50	7,0 – 7,5
	APM ₂	0,16 – 3,0	Tris : 20 – 100	7,0 – 7,5
AAP	AAP ₁	5,0 – 250	Tris : 10 – 50	7,5 – 8,5
API	API ₁	4,0 – 185	Tris : 10 – 50	7,5 – 8,5
LAP	LAP ₁	4,0 – 6,0	Tris : 20 – 100	7,5 – 8,5
	LAP ₂	13 – 325	Tris: 10 – 50	7,5 – 8,5

Tabelle 16: Sequenzierungsversuche mit Amino-peptidase-Kombinationen

Gruppen Bezeichnung	Experiment Bezeichnung	angewendeter Aktivitätsbereich [mU]	Puffersalz- konzentration [mM]	pH Bereich
sb(AP)	API + APM	185 + 0,05	Tris : 20 – 100	7,0 – 7,5
	APM + AAP	(0,2 – 2,0) + (5,0 – 10)	Tris : 20 – 100	7,0 – 7,5
	AAP + APM	(5,0 – 25) + (0,08 – 0,2)	Tris: 20 – 100	7,5 – 8,5
	AAP + LAP	250 + 325	Tris : 20 – 100	7,5 – 8,5
	APM + LAP	(0,08 – 2,0 + (13 – 325)	Tris : 20 – 100	7,5 – 8,5
pb(AP)	APM + AAP	(0,16 – 0,4) + (10 – 50)	Tris : 10 – 50	7,5 – 8,5
pt(AP)	APM + AAP + API	0,08 + 2,0 + 0,8	Tris: 10 – 50	7,5 – 8,5

C-terminale Sequenzierungen:**Tabelle 17: Sequenzierungsversuche mit einzelnen Carboxypeptidasen**

Gruppen Bezeichnung	Experiment Bezeichnung	angewendeter Aktivitätsbereich [mU]	Puffersalz- konzentration [mM]	pH Bereich
CPY	CPY ₁	0,15 – 30	NaCit: 0,1 – 13	4,5 – 6,0
	CPY ₂	2,4 – 12	NaCit: 1 – 5	4,5 – 6,0
CPP	CPP ₁	0,025 – 0,5	NaCit: 0,1 – 3	4,5 – 6,0
	CPP ₂	0,1 – 1,0	NaCit: 0,5 – 5	4,5 – 6,0
CPW	CPW ₁	1,3 – 5,3	NaCit: 0,1 – 0,5	4,2 – 6,0
	CPW ₂	3,9 – 7,9	NaCit: 0,4 – 0,8	4,2 – 6,0
CPA	CPA ₁	0,5 – 8,8	Tris : 5 – 25	7,0 – 8,5
CPB	CPB ₁	1,5 – 7,5	Tris: 5 – 25	7,0 – 8,5

Tabelle 18: Sequenzierungsversuche mit Carboxypeptidase-Kombinationen

Gruppen Bezeichnung	Experiment Bezeichnung	angewendeter Aktivitätsbereich [mU]	Puffersalz- konzentration [mM]	pH Bereich
sb(CP-I)	CPY + CPP	(0,3 – 6,0) + (0,025 – 0,5)	NaCit: 0,3 – 5	4,5 – 6,0
	CPP + CPY	(0,050 – 0,5) + (1,5 – 6,0)	NaCit: 0,9 – 5	4,5 – 6,0
sb(CP-II)	CPB + CPP	(1,9 – 3,8) + (0,050 – 0,5)	Tris : 5 – 25 NaCit: 0,2 – 3	5,0 – 7,0
	CPB + CPY	(1,9 – 3,8) + (0,60 – 6,0)	Tris : 5 – 25 NaCit : 0,4 – 3	5,0 – 7,0
	CPB + CPW	1,9 + 0,53	Tris : 5 – 25 NaCit : 0,1	4,5 – 7,0
	CPB + CPA	1,9 + 2,2	Tris : 10 – 50	7,0 – 8,5
sb(CP-III)	CPY + CPA	3,0 + (2,2 – 8,8)	NaCit: 1,25 Tris: 5 – 25	5,0 – 7,0
	CPY + CPW	6,0 + (2,6 – 5,3)	NaCit: 4	4,5 – 6,0
	CPA + CPP	2,2 + 0,125	Tris : 5 – 25 NaCit : 0,6	5,0 – 7,0
pb(CP)	(CPY+CPP)	(0,3 + 0,025) – (3,0 + 0,25)	NaCit : 0,1 – 3	4,5 – 6,0
spt(CP)	CPW + (CPY+CPP)	(1,3) + (3,0 + 0,25)	NaCit : 2	4,5 – 6,0
	(CPY+CPP) + CPW	(3,0 + 0,25) + (1,3 – 2,6)	NaCit : 2	4,5 – 6,0
	(CPY + CPP + CPW)	(1,3 + 0,87 + 0,86)	NaCit : 2,6	4,5 – 6,0
	(CPY + CPP + CPA)	(1,8 + 0,88 + 13)	NaCit: 1,6 Tris: 3	5,0 – 7,0

Bei der Anwendung nur einer Peptidase für die Sequenzierung besteht innerhalb einer Gruppe nur eine Unterscheidung nach der Reaktionszeit – dargestellt durch den tiefgestellten Index. Dabei bedeutet:

- Index 1: Reaktionszeit 1 – 6 min
- Index 2: Reaktionszeit 6 – 12 min

Bei der Anwendung von Peptidasekombinationen wurde eine differenziertere Unterscheidung vorgenommen. Bei den Gruppenbezeichnungen bedeuten:

- s: serielle / nacheinanderfolgende Auftragung der genannten Enzyme
- p: parallele Auftragung / Auftragung als Mischung
- b: binär; Kombination aus zwei Enzymen
- t: ternär; Kombination aus drei Enzymen

Bei den Carboxypeptidasen wurden zweckmäßiger Weise Kombinationen der unspezifischen Peptidasen CPY, CPP, CPW und CPA untereinander (CP-I und CP-III) unterschieden von Enzymkombinationen aus der spezifischen Peptidase CPB mit den unspezifischen Peptidasen (CP-II). Folgende Intervalle von Reaktionszeiten wurden bei den Enzymkombinationen untersucht:

- seriell-binäre Auftragung: 4 – 12 min
- parallel-binäre und parallel ternäre Auftragung: 2 – 12 min
- seriell / parallel-ternäre Auftragung: 4 – 12 min

Bei den *on-target* Sequenzierungen und den Sequenzierungen auf der Membran hängen die genauen Reaktionszeiten natürlich sehr stark von folgenden Parametern ab:

- Reaktionstemperatur
- Reaktionsvolumen

Die untersuchten Spaltungstemperaturen lagen in einem Intervall von 25 – 45°C. Da die Aktivitäten temperaturabhängig sind, dienen die angegebenen Aktivitätsbereiche der Peptidasen lediglich als Anhaltspunkt für das untersuchte Temperaturintervall. Da z.B. bei APM die

Aktivität bei 25°C nur 40% der angegebenen Aktivität bei 37°C entspricht, sollte je nach betrachteter Peptidase eine Schwankungsbreite von mindestens $\pm 50\%$ der angegebenen Aktivitätswerte in den obigen Tabellen einkalkuliert werden. Das Reaktionsvolumen, also das aufgetragene Volumen der Enzymlösungen, liegt im Bereich zwischen 0,8 – 1,2 µl. Der Versuchsparameter „Luftfeuchtigkeit“ richtet sich nach den jeweiligen Umgebungsbedingungen. Dieser Parameter zeigt jedoch in seinem Schwankungsbereich in den Versuchen keinen ausgeprägten Einfluss auf die Reaktionszeiten.

Die folgenden Tabellen zeigen, welche Spaltungsexperimente bei den Sequenzierungen der Proteine in einzelnen getestet wurden. Bei Experimenten mit Einzelpeptidasen erschienen CPY und CPP auf C-terminaler Seite sowie APM auf N-terminaler Seite im anfänglichen Verlauf der Versuche die größten Erfolgchancen zu bieten. Dementsprechend wurden mit diesen Einzelpeptidasen wesentlich mehr Versuche durchgeführt. Auf der C-terminalen Seite zeigte auch CPW ein gutes Potential. Diese Peptidase stand jedoch nicht von Anfang an zur Verfügung, weshalb von diesem Enzym weniger Sequenzierungsergebnisse zur Beurteilung zur Verfügung stehen. Ebenso verhält es sich mit den Aminopeptidasen AAP und API, die ebenfalls erst bei späteren Versuchen verfügbar waren. CPA bei den Carboxypeptidasen und LAP bei den Aminopeptidasen zeigten nach den ersten Versuchen keine befriedigenden Resultate gegenüber der Anwendung anderer Exopeptidasen. Sie wurden daher als Einzelpeptidasen weniger intensiv für Sequenzierungsstudien herangezogen. Stattdessen erfolgte die Verwendung dieser Peptidasen überwiegend in Form von Peptidasemischungen. Die Carboxypeptidase CPB wurde im wesentlichen in serieller Kombination mit den anderen unspezifischen Carboxypeptidasen in Sequenzierungen der Gruppe sb(CP-II) bei der Untersuchung von Spaltpeptiden aus Endoproteinasespaltungen mit Trypsin oder LysC angewendet.

Tabelle 19: Anwendung einzelner Carboxypeptidasen auf Proteinspaltungen

	CPY		CPP		CPW		CPA
	CPY-I	CPY-II	CPP-I	CPP-II	CPW-I	CPW-II	CPA-I
LysC							
β-Lactoglobulin	X						
β-Casein							
Myoglobin	X						
Aldolase	X		X	X			X
Cytochrom C	X		X				
<i>DrTI (Proteinase Inhibitor)</i>	X		X		X		
Rinderserum Albumin							
GluC							
α-Casein	X						
β-Lactoglobulin	X	X					
Aldolase	X	X	X	X			
Trypsin							
Myoglobin	X		X				
Aldolase	X		X				
Schweineserum Albumin					X		
Chymotrypsin							
Aldolase	X	X	X	X			X
Alkoholdehydrogenase	X	X	X	X			X
AspN							
<i>DrTI (Proteinase Inhibitor)</i>	X	X	X		X	X	
Summe	12		9		4		3

Tabelle 20: Anwendung von Carboxypeptidase-Kombinationen auf Proteinspaltungen

	seriell, binär			parallel, binär	ternär
	sb(CP-I)	sb(CP-II)	sb(CP-III)	pb(CP)	spt(CP)
LysC					
β-Lactoglobulin	X				
β-Casein	X				
Myoglobin					
Aldolase		X	X		
Cytochrom C		X			
<i>DrTI (Proteinase Inhibitor)</i>	X			X	X
Rinderserum Albumin	X				
GluC					
α-Casein	X				
β-Lactoglobulin					
Aldolase	X			X	
Trypsin					
Myoglobin	X	X		X	
Aldolase	X			X	
Schweineserum Albumin		X			
Chymotrypsin					
Aldolase	X			X	X
Alkoholdehydrogenase	X		X		
AspN					
<i>DrTI (Proteinase Inhibitor)</i>	X		X		
Summe	13			5	2

Tabelle 21: Anwendung einzelner Aminopeptidasen auf Proteinspaltungen

	APM		AAP	API	LAP	
	APM-I	APM-II	AAP-I	API-I	LAP-I	LAP-II
LysC						
β-Lactoglobulin	X					
β-Casein	X					
Myoglobin	X	X				
Aldolase	X	X	X	X	X	
Cytochrom C						
<i>DrTI (Proteinase Inhibitor)</i>	X		X	X		
Rinderserum Albumin						
GluC						
α-Casein	X					
β-Lactoglobulin	X	X				
Aldolase	X		X	X		
Trypsin						
Myoglobin	X		X	X		
Aldolase	X		X	X		
Schweineserum Albumin	X		X	X		
Chymotrypsin						
Aldolase	X	X	X	X	X	X
Alkoholdehydrogenase						
AspN						
<i>DrTI (Proteinase Inhibitor)</i>	X					
Summe	13		7	7	2	

Tabelle 22: Anwendung von Aminopeptidase-Kombinationen auf Proteinspaltungen

	seriell, binär sb(AP)	parallel, binär pb(AP)	parallel, ternär pt(AP)
LysC			
β-Lactoglobulin			
β-Casein			
Myoglobin			
Aldolase	X		
Cytochrom C			
<i>DrTI (Proteinase Inhibitor)</i>			
Rinderserum Albumin	X		
GluC			
α-Casein			
β-Lactoglobulin			
Aldolase	X		
Trypsin			
Myoglobin	X		X
Aldolase	X	X	
Schweineserum Albumin		X	
Chymotrypsin			
Aldolase	X		
Alkoholdehydrogenase			
AspN			
<i>DrTI (Proteinase Inhibitor)</i>			
Summe	6	2	1

2.6 MALDI-TOF-MS Messungen

2.6.1 Parameter der UV-MALDI-MS Messung von Peptiden

Die im folgenden beschriebenen Parametersätze wurden für die UV-MALDI-MS Messung aller peptidhaltigen Proben in dieser Arbeit angewendet. Dazu zählen die über Endoproteinasen erzeugten, komplexen Spaltgemische, die HPLC-getrennten Peptide dieser Spaltgemische, sowie die Peptidleitern und auch die im späteren noch beschriebenen Peptidderivate. Bei den UV-MALDI-MS Messungen von PVDF-Blot Membranen wurde ebenfalls die nachfolgenden Spannungsparameter eingesetzt. Dabei handelt es sich um Parametersätze, die für die Peptidmessungen empirisch im Hinblick auf Auflösung und Signalintensität optimiert wurden. Alle Messungen wurden im positiven Reflektron-Modus durchgeführt, sowie unter Verwendung von verzögerter (gepulster) Ionenextraktion (*delayed ion extraction* = DE oder *pulsed ion extraction* = PIE, siehe oben).

Einzelne Parameter, wie z.B. die Detektorspannung müssen an die jeweilige Messung angepasst werden, da hier Faktoren wie die Menge und die molare Response des Peptids aber auch das Alter der Detektor-MCPs eine Rolle spielen. Auch bei der Einstellung des Deflektors hängt die Zeitspanne der angelegten Deflektionsspannung (+2 kV) von individuellen Parametern ab, wie z.B. der Art der verwendeten Matrix, anwesenden Puffersalzen oder überschüssigen Reagenzien in der Probe. Minimal wurde der Deflektor auf einen Wert von 400 Da eingestellt, womit die Deflektion der Matrixsignale aller verwendeten Matrices möglich ist. Für die Abschwächung des Lasers kann ebenfalls kein absoluter Wert oder Wertebereich angegeben werden. Die Laserintensität für die Messungen wurde in der Regel ca. 5 – 30% über dem Schwellenwert gewählt, der zur initialen Erzeugung und Detektion von Analytionen erforderlich war.

Folgende Parametersätze wurden bei Messungen von Peptiden mit den Matrices CHCA, DHB und DHBs-Matrix verwendet:

Parametersatz 1

Spannung an P1 (U_{IS1}):	20 kV
Spannung an P2 (U_{IS2}):	15,4 kV
Linsenspannung: U_{Linse} :	7,2 kV
Spannung an Reflektron (U_{ref}):	21,5 kV
Detektorspannung Reflektormodus ($U_{det,ref}$):	1,3-1,5 kV

Dieser Parametersatz wurde für die anfänglichen Messungen dieser Arbeit verwendet. Er weist leichte Variationen in den Spannungen für P2 und für Linse in verschiedenen Experimenten auf, je nachdem ob mit CHCA oder DHB gemessen wurde. U_{IS2} kann hier dann bis zu 16,6 kV betragen, U_{Linse} bis zu 6,7 kV. Parametersatz 2 kam im späteren Verlauf der Arbeit zum Einsatz und spiegelt eine leichte Optimierung für Auflösung und Empfindlichkeit wieder.

Parametersatz 2

Spannung an P1 (U_{IS1}): 20,0 kV
Spannung an P2 (U_{IS2}): 16,0 kV
Linsenspannung: U_{Linse} : 6,7 kV
Spannung an Reflektron (U_{ref}): 22,8 kV
Detektorspannung Reflektormodus ($U_{det,ref}$): 1,3-1,5 kV

Messungen von der PVDF-Membran wurden einerseits mit Parametersatz 1 durchgeführt, u.a. die Messung der Dynorphin-Peptide von der Membran, zum Teil wurde jedoch auch mit höheren Beschleunigungsspannungen und größerer Potentialdifferenz gemäß Parametersatz 3 gearbeitet (höhere Beschleunigungsenergien):

Parametersatz 3

Spannung an P1 (U_{IS1}): 28,5 kV
Spannung an P2 (U_{IS2}): 21,3 kV
Linsenspannung: U_{Linse} : 9,9 kV
Spannung an Reflektron (U_{ref}): 30,0 kV
Detektorspannung Reflektormodus ($U_{det,ref}$): 1,3-1,5 kV

2.6.2 Parameter der UV-MALDI-MS Messung von Proteinen

Die MALDI-MS Messungen von Proteinproben, die in Lösung wie unter 2.4.2 nach der *dried-drop* Methode mit DHB präpariert wurden, erfolgte im linearen Modus unter Verwendung der PIE. Da bei den Proteinmessungen grundsätzlich nur Analyten mit einer Masse über 10 kDa interessierten, wurden allgemeinen Moleküle mit einer Masse bis 1000 Da durch Deflektion ausgeblendet. MALDI-MS Messungen von Proteinen wurden im Rahmen dieser Arbeit meist nur zum Zweck der Reinheits- und Reaktionskontrolle durchgeführt. Dabei wurde molare Masse der Proteine vor und nach der Proteinalkylierung überprüft. Folgende Parameter wurden bei Messungen von Proteinen mit DHB-Matrix verwendet:

Parametersatz 4:

Spannung an P1 (U_{IS1}): 24,0 kV
Spannung an P2 (U_{IS2}): 15,7 kV
Linsenspannung: U_{Linse} : 8,5 kV
Detektorspannung Reflektormodus ($U_{det,lin}$): 1,4-1,9 kV

2.6.3 Parameter der IR-MALDI-MS Messungen von Peptiden

Sämtliche MALDI-Messungen von Peptiden unter Verwendung von Infrarotlaserlicht der Wellenlänge 2,94 μm wurden, wie die entsprechenden Messungen mit ultraviolettem Laserlicht, im Reflektormodus mit PIE ausgeführt. Folgender Parametersatz wurde für die IR-MALDI-MS verwendet:

Parametersatz 5

Spannung an P1 (U_{IS1}): 20,0 kV
Spannung an P2 (U_{IS2}): 15,0 kV
Linsenspannung: U_{Linse} : 7,2 kV
Spannung an Reflektron (U_{ref}): 21,5 kV
Detektorspannung Reflektormodus ($U_{det,ref}$): 1,3-1,5 kV

2.6.4 Kalibrierung

Kalibrierungen für die MALDI-TOF-MS Messung einzelner Peptide, kompletter Spaltpeptidmischungen nach der Endopeptidasespaltung oder der Peptidleitern erfolgten in dieser Arbeit nahezu ausschließlich extern. Für externe Kalibrierung im Bereich der analysierten Peptide und Leiterpeptide wurde eine Mischung der nachfolgend aufgelisteten Standardpeptide verwendet. Da die Kalibrierung bei MALDI-MS nicht quantitativ erfolgt, wurde die Konzentration der Kalibranden empirisch so eingestellt, dass alle Kalibranden unter den gegebenen Messbedingungen eine möglichst optimale Signalintensität für die gewählte Präparation aufweisen. Die Präparation der Kalibrandenmischung erfolgte nach Standardmethode aus 2.4. Die Konzentrationen aller Peptide in den so angewendeten Kalibrieremischungen lag zwischen 0,01 und 1,0 pmol/ μl . Für die MALDI-Präparation der Kalibranden wurde jeweils diejenige Matrix verwendet, die auch für die nachfolgende Messung der Analyten verwendet wurde. Ebenso wurde bei den Spannungen für die Messung der Kalibranden der gleiche Parametersatz gewählt, wie für die anschließende Messung der zu

bestimmenden Peptide. Zur Erstellung der Kalibrierfunktion wurden die monoisotopischen Massen verwendet.

Tabelle 23: Kalibranden für den Peptidmassenbereich

Kalibrand	Kalibriermasse $[M+H]^+$ (monoisotopisch)
Angiotensin 2	1046,5423
Bradykinin	1060,5693
Angiotensin 1	1296,6853
Substance P	1347,7361
Neurotensin	1672,9176
ACTH Fragment 18-39	2465,1989
Insulin B-Kette, oxidiert (Rind)	3494,6513
Insulin (Rind)	5731,6165

Es wurde immer nur auf den interessierenden Bereich kalibriert – außerhalb liegende Massen der Kalibrier Mischung wurden nicht berücksichtigt. Die Kalibrierung auf zwei Massen erfolgte durch lineare Regression über die XACQ-Software. Bei der Verwendung von drei oder mehr Referenzmassen wurde eine quadratische Regression gewählt, da einführende Vergleichsmessungen hier eine höhere Massengenauigkeit zeigten. Nur zur Messung kompletter Gemische aus Trypsinspaltungen erfolgte die Kalibrierung intern über eine lineare Zweipunkt-Kalibrierung auf folgende Signale von Autolyseprodukten des Trypsins:

- ❖ Fragment 64-69: $[M+H]^+_{\text{mono}} = 659,3840$ Da (Sequenz: SGIQVR);
alternativ: Fragment 112-119: $[M+H]^+_{\text{mono}} = 805.4168$ Da (Sequenz: SAASLNSR)
- ❖ Fragment 70-89: $[M+H]^+_{\text{mono}} = 2163,0570$ (Sequenz: GEDNINVVEGNEQFISASK)

2.6.5 Datenakquisition und Datenauswertung

Für alle Messungen (Proteine, proteolytische Spaltgemischen von Proteinen, einzelnen Peptidfraktionen aus proteolytischen Spaltgemischen und Peptidleitern) wurden allgemein zwischen 50 und 200 Einzelspektren summiert. Die Lokalisation von geeigneten Stellen für die Desorption und das Anfahren der selektierten Spots erfolgt in diesem Teil der Arbeit zunächst grundsätzlich manuell (Selektion visuell über CCD Kamera und Monitor). Um mögliche Unregelmäßigkeiten der Analytverteilung in der Matrix zu berücksichtigen, wurden die Einzelspektren von mindestens drei bis fünf verschiedenen Stellen der Präparation gesammelt. Im Falle der Infrarot MALDI-MS ist der Abtrag der Probe am Ort des

Laserbeschusses sehr hoch. Hier wurden zum Teil nur 10 – 25 Spektren akkumuliert und der Ort der Messung jeweils bereits nach ein bis zwei Laserschüssen variiert.

Die summierten Spektren wurden entweder direkt auf der Sun-Station mit dem Programm XMASS, oder auf dem PC mit dem Programm „Bruker Data Analysis“ ausgewertet, wobei in beiden Fällen keine Prozessierung (Glättung) der Daten erfolgte. Die in dieser Arbeit vorliegenden Daten basieren auf Massen, die von der Software über den Centroid bei 80% der maximalen Peakhöhe errechnet wurden.

2.7 Derivatisierungen

Alle nachfolgend beschriebenen Derivatisierungen wurden zunächst in Lösung durchgeführt, später jedoch zum Teil auch direkt auf dem MALDI-Target. Eingangsversuche mit synthetischen Peptiden erfolgten mit 1nmol bis zu 250fmol Ausgangsmaterial. Bei Derivatisierungen proteolytischer Spaltfragmente nach der HPLC-Trennung sind die genauen Mengen des derivatisierten Peptids unbekannt. Quantitative Bestimmungen einzelner HPLC-Fractionen zeigen jedoch, dass die Konzentrationen der derivatisierten Peptide unter 10pmol/ μ l liegen – zumeist war jedoch die Peptidmenge für eine Bestimmung zu gering und die Konzentrationen liegen in solchen Fällen vermutlich unter 1pmol/ μ l. Da sich alle Peptide zunächst in wässriger 0,1%iger TFA-Lösung befanden, wurde in Vorversuchen verifiziert, dass der pH-Wert durch die verwendete Pufferlösung bei der Reaktion in den für die Derivatisierung erforderlichen Bereich verschoben wurde.

2.7.1 Peptidderivatisierung mit Säurechloriden und Succinimidylcarbonaten

Die im folgenden beschriebene Vorschrift für die Peptidderivatisierung fand allgemein für Umsetzungen mit Säurechloriden oder Succinimidylcarbonaten als Reaktionspartner Anwendung. Für die Derivatisierung wurde 1 μ l der Peptidlösung im 0,5ml PP-Reaktionsgefäß mit 20 μ l einer 10mM Natriumbicarbonatlösung pH 9,2 versetzt. Anschließend wurden 10 μ l der Säurechlorid- oder Succinimidylcarbonatlösung in einer Konzentration von 2,0mg/ml in Aceton zugegeben und so die Umsetzung gestartet. Die optimierten Reaktionsbedingungen der unterschiedlichen Säurechloride sind in der nachfolgenden Tabelle zusammengefasst.

Tabelle 24: Reaktionstemperaturen und –zeiten bei der Peptidderivatisierungen mit Säurechloriden und Succinimidylcarbonaten

Name (verwendete Abkürzung)	Reaktionstemperatur	Reaktionszeit
Dansylchlorid (<i>DNS-Cl</i>)	37°C	60min
Dabsylchlorid (<i>DABS-Cl</i>)	70°C	20min
Sulforhodamin B Säurechlorid (<i>SRB-Cl</i>)	30°C	60min
6-Bromhexanoylchlorid (<i>6-BH-Cl</i>) 3-Brombenzoylchlorid (<i>3-BB-Cl</i>) 4-Brombenzoylchlorid (<i>4-BB-Cl</i>) (9-Fluorenylmethyl)-chloroformiat (<i>FMOC-Cl</i>)	40°C	60min
(9-Fluorenylmethyl)-succinimidylcarbonat (<i>FMOC-NHS</i>) N-(2-Brombenzyloxy-carbonyl)-hydroxysuccinimid (<i>BBOC-NHS</i>)	40°C	60min

2.7.2 Peptidderivatisierung mit Isothiocyanaten

Derivatisierungen mit Eosin 5 Isothiocyanat (E5-ITC) und Rhodamin B Isothiocyanat (RB-ITC) wurden wie folgt durchgeführt: 1 µl der Peptidlösung wurde im 0,5 ml PP-Reaktionsgefäß mit 10 µl einer 50 mM N-Methylpiperidinlösung in DMSO versetzt. Anschließend wurden 50 µl der Isothiocyanatlösung in einer Konzentration von 2,0 mg/ml in DMSO zugegeben und so die Umsetzung gestartet. Die optimale Reaktionstemperatur lag zwischen 55 und 70°C, die Reaktionszeit betrug 75min.

2.7.3 Peptidderivatisierung mit 6-Aminochinolyl-N-hydroxysuccinimid

Die Vorschrift für die Derivatisierung mit 6-Aminochinolyl-N-hydroxysuccinimid (ACQ™) ist abgewandelt nach [Cohen 1994, De Antonis 1994]. Es wurde 1 µl der Peptidlösung mit 20 µl eines wässrigen 100mM Natriumboratlupuffers (pH 8,8) versetzt. 20 µl einer Lösung des ACQ in Acetonitril wurden zu der Peptidlösung zugegeben und die Reaktion bei 40°C für 60min durchgeführt.

2.7.4 Peptidderivatisierung durch Acetylierung

Die Derivatisierungen wurden mit einer Mischung aus Essigsäureanhydrid/Eisessig in Aceton in einem 0,5ml PP-Reaktionsgefäß durchgeführt. Zu 1,0 µl der Peptidlösung wurden 40 µl Aceton gegeben. Anschließend wurden 10 µl eine Mischung aus Essigsäureanhydrid/Eisessig 50/50 (v/v) zugegeben. Nach 60min bei Raumtemperatur (27°C) wurde der Reaktionsansatz mit 20 µl Acetonitril/Wasser 50/50 + 0,1% TFA verdünnt und in der *Speed-vac* bis zur

Trockene eingeeengt. Danach wurde das Reaktionsprodukt in 4µl Acetonitril/Wasser 50/50 + 0,1% TFA aufgenommen.

2.7.5 HPLC-Reinigung der Peptidderivate

Sofern eine enzymatische Sequenzierung eines Peptidderivats erfolgte, musste das Derivat in einigen Fällen (siehe Ergebnisteil) über eine *reversed-phase* HPLC von überschüssigem Reagenz und Puffersalzen befreit werden. Diese Reinigung erfolgte mit Hilfe einer C18-AQ Reprosil-Pur Säule mit 5µm Partikelgröße, 125/1mm auf dem HPLC-System 2 (Pharmacia-LKB Smart System) unter Verwendung des folgenden Gradienten:

20%B $\xrightarrow{10\text{min}}$ 20%B $\xrightarrow{30\text{min}}$ 75%B $\xrightarrow{10\text{min}}$ 85%B $\xrightarrow{10\text{min}}$ 85%B

Eluent A: Wasser + 0,1% TFA

Eluent B: Acetonitril + 0,09% TFA

Fluss: 60µl/min

Als Detektionswellenlänge wurde, falls möglich, neben der unspezifischen Absorptionswellenlänge von 210nm die spezifische Absorptionswellenlänge einer chromophoren Gruppe des Derivatisierungsreagenzes verwendet.

Tabelle 25: Spezifische Detektionswellenlängen bei der HPLC-Reinigung der Peptidderivate

Name	verwendete Abkürzung	Detektionswellenlänge [nm]
dansylierte Peptide	<i>DNS-Peptid</i>	340
dabsylierte Peptide	<i>DABS-Peptid</i>	436
Sulforhodamin B-Peptid	<i>SRB-Peptid</i>	556
6-Bromhexanoyl-Peptid	<i>6-BH-Peptid</i>	206
3-Brombenzoyl-Peptid	<i>3-BB-Peptid</i>	254
4-Brombenzoyl-Peptid	<i>4-BB-Peptid</i>	254
9-Fluorenylmethyloxycarbonyl-Peptid	<i>Fmoc-Peptid</i>	266
N-(2-Brombenzyloxycarbonyl)-Peptid	<i>BBOC-Peptid</i>	254
Rhodamin B-Peptid	<i>RB-Peptid</i>	556
Eosin 5-Peptid	<i>E5-Peptid</i>	521
ACQ™-Peptid	<i>ACQ-Peptid</i>	250

3 Ergebnisse

3.1 Synthetische Peptide

Für erste Untersuchungen der Exopeptidasen wurden ausschließlich synthetische Peptide herangezogen, deren Massen mit ca. 1100 Da bis 2800 Da in dem für die Sequenzierung optimalen Bereich liegen. Die Auswertungen der resultierenden Leiterspektren aus den Sequenzierungen beziehen sich ausnahmslos auf monoisotopische Massen.

Die Sequenzierungen mit den verschiedenen Peptidasen wurden eingangs in Lösung ausprobiert und anschließend für alle Peptide *on-target* durchgeführt. Die sequenzierte Absolutmenge der Peptide wurde im Bereich von 50 bis 500 fmol variiert. Die Peptide wurden dazu jeweils aus unterschiedlich konzentrierten Lösungen in ACN/Wasser 50/50 + 0,1% TFA aufgetragen. Der Abbau erfolgte bei 27 – 30°C. Die MALDI-MS Messung der erzeugten Peptidleitern erfolgte mit Hilfe von DHB als Matrix und UV-Laserlicht.

3.1.1 Sequenzen

Für die in Tabelle 26 aufgelisteten Sequenzen der 21 untersuchten synthetischen Peptide ist der insgesamt aus den Leiterspektren ausgelesene Sequenzteil für jedes Peptid fettunterstrichen dargestellt. Sequenzteile, die in eckigen Klammern [] stehen, deuten eine Lücke („Gap“) beim Auslesen der Sequenz an, d.h. durch eine ungünstige Abspaltungskinetik der Aminosäuren lässt sich ihre Abspaltung nur in der Summe detektieren.

Tabelle 26: Resultate der enzymatischen Sequenzierungen einiger synthetischer Peptide

Peptid	Sequenz	[M+H] ⁺ monoisotopisch in Da
Dynorphin A	<u>YGGFLRRIRPKLKWDNQ</u> YGGFLRRIRP <u>KLKWDNQ</u>	2147,1992
P12-1395	REVHTN <u>QDPLDA</u> <u>REVHTN</u> QDPLDA	1394,6664
Angiotensin 1	<u>DRVYIHPFHL</u> DRVYIH <u>PFHL</u>	1296,6853
Angiotensin 2	¹ DRVYIH <u>PF</u>	1046,5423
Substance P	¹ RPKPQQ <u>FFGLM-NH2</u>	1347,7361
P18-2107	<u>KLMDLDVEQLGI</u> PEQEYS KLMDLD <u>VEQL [GIP] EQEYS</u>	2107,0270

Peptid	Sequenz	[M+H] ⁺ monoisotopisch in Da
ACTH (18-39)	RPVKVYPNGAEDESAEAFPLEF RPVKVYP <u>NGAEDESAEAFPLEF</u>	2465,1989
P24-2836	TIRNSGLRNIQPRFAKNPHLRYI TIRNSGLRNIQPRFAKN <u>PHLRYI</u>	2835,5970
P24-2768	TVVDSGLRFVSRQAFVKINLQYI TVVDSGLRFVSRQAFVKINLQYI	2767,5259
P24-2722	<u>FEQ</u> RVNSDVLTVSTVNSQDQVTQK FEQRVNSDVL <u>[TVSTVNS] QDQVTQK</u>	2722,3646
Neurotensin	(pyro) ELYENKPRRPYIL (pyro) ELYENKPRRPYIL	1672,9176
P31-3467	YSDMREANYIGSDKYFHARGNYDAAKRGPGG YSDMREANYIGSDKYFHARGNYDAAKRGPGG	3648,8392
P12-1303	<u>AGHEY</u> GAEALER AGHEYG <u>AEALER</u>	1302,6078
P15-1812	<u>IKVTLVFEH</u> VDQDLR IKVTLVFEH <u>VDQDLR</u>	1812,0020
P10-1213	<u>FLDTLVV</u> LHR FLDTLV <u>VLHR</u>	1212,7105
P9-1069	¹ TKNYKQTS <u>V</u>	1068,5690
P10-1191	¹ NKDKNQESD <u>I</u>	1191,5653
P10-1360	¹ NKKKKKEY <u>FF</u>	1359,7790
P14-1713	¹ ESKFQQKL <u>AEFTTR</u>	1712,8973
P14-1624	¹ ISSYQDAIEIE <u>LEN</u>	1623,7754
P14-1639	¹ REDFSGLLP <u>EEFIS</u>	1638,8016

¹ Eine N-terminale Sequenzierung wurde mit diesem Peptid nicht durchgeführt.

3.1.2 Abbaulängen, Sequenzlängen und Sequenzabdeckung

Die folgende Tabelle zeigt im Detail, welche Sequenzierungsexperimente mit den synthetischen Peptiden durchgeführt wurden und wie sich der Sequenzierungserfolg über die einzelnen Experimente verteilt:

Tabelle 27: Statistik der N- und C-terminalen Sequenzierungen einiger synthetischer Peptide

Abbauart	Sequenzabdeckung		Mittelwert Abbaulänge	Mittelwert Sequenzlänge
	Abbau	Sequenz		
N-terminal				
APM	21%	18%	4	3-4
AAP	32%	32%	4	4
API	49%	38%	6	4-5
Mittelwert	34%	29%	4-5	4
C-terminal				
CPY	40%	33%	5-6	4-5
CPP	28%	19%	4	3-4
CPW	27%	27%	3-4	3-4
sb(CP-I)	30%	30%	5-6	5-6
Mittelwert	31%	27%	4-5	4

Unterschieden wurde hier, wie auch später bei den Sequenzierungen der proteolytischen Spaltpeptide, aus praktischer Sicht zwischen der **Abbaulänge** und der **Sequenzlänge**. Der Begriff der „Abbaulänge“ bezieht sich auf alle abgebauten Aminosäuren ungeachtet der Tatsache, dass bei der Sequenzierung auch Lücken, z.B. durch eine ungünstige Abbaukinetik, vorhanden sein können. Bei Sequenzlücken im Leiterspektrum kann meist weder die Art noch die Abfolge der abgebauten Aminosäuren ohne weiteres festgestellt werden. Daher ist die Abbaulänge allein kein Maß, ein Experiment mit einer bestimmten Peptidase oder Peptidasekombination bezüglich seiner Eignung für die Leitersequenzierung zu beurteilen. Hierzu ist der in dieser Arbeit verwendete Begriff der „Sequenzlänge“ besser geeignet. Die Sequenzlänge bezeichnet die Zahl der Aminosäuren, für die sich aus dem Leiterspektrum tatsächlich eine Sequenzabfolge festlegen lässt. So gilt als beispielsweise für die Abbau- und Sequenzlängen bei P18-2107 aus Tabelle 26:

Sequenzierung	Abbaulänge (Aminosäuren)	Sequenzlänge (Aminosäuren)
N-terminal	12	12
C-terminal	12	9
Summe (C-terminal und N-terminal)	18	18

Das Prolin liegt zwar in einer Sequenzlücke, rechnerisch erlauben die Resultate der C- und N-terminalen Sequenzierungen jedoch einen Rückschluss auf Prolin als fehlende Aminosäure.

Im Hinblick auf die Sequenzierung eines Peptids oder eines Proteins über seine Spaltfragmente ist letztlich die erhaltene Sequenzlänge von entscheidender Bedeutung. Die Abbaulänge ist allerdings für das Abschätzen des Potentials einer untersuchten Peptidase oder Peptidasekombination für die Sequenzierung ebenfalls ein wichtiges Kriterium. Bei großer Abbaulänge aber geringer Sequenzlänge ist zu testen, ob nicht durch eine geeignete Kombination mit einer anderen Peptidase oder eine Abänderung der Reaktionsbedingungen eine verbesserte Kinetik und somit ein lückenloseres Leiterspektrum erhalten werden kann.

Der Begriff der **Sequenzabdeckung** in Tabelle 27 bezieht sich auf das Verhältnis der Zahl abgebauter (linke Reihe in der Spalte „Sequenzabdeckung“) beziehungsweise sequenzierter (rechte Reihe in der Spalte „Sequenzabdeckung“) Aminosäuren zur gesamten Sequenzlänge. Der Wert für die Sequenzabdeckung ist somit unabhängig von der Abbau- oder Sequenzlänge. Betrachtet man, wie oben geschehen, eine umfassende Zahl von Peptiden, so lässt sich über den errechneten Mittelwert der Sequenzabdeckung durchaus abschätzen, wie groß letztlich die zu erwartende Sequenzabdeckung bei einer Proteinsequenzierung über proteolytische Spaltfragmente sein wird.

Die erhaltenen Sequenzen der einzelnen Experimente in Tabelle 26 sind zum Teil komplementär. Das heißt z.B., dass der Abbau eines bestimmten Peptids mit einer Amino-peptidase die N-terminale Sequenz der Aminosäuren 1 - 4 ergibt, während der Abbau des gleichen Peptids mit einer anderen Amino-peptidase die N-terminale Sequenz der Aminosäuren 3 - 6 ergibt. Die Sequenzlänge beträgt damit zwar in beiden Fällen 4, für die Summe der beiden Spaltungen ergibt sich aber eine Sequenzlänge von 6. Betrachtet man die Summe der Sequenzinformation bei Peptiden, die mit verschiedenen Einzelpeptidasen oder Peptidasekombinationen in unabhängigen Experimenten erhalten wurde, so ergibt sich bezüglich der Abbau- und Sequenzlängen (und damit letztlich der Sequenzabdeckungen) zumeist ein wesentlich besser Wert als der oben berechnete Mittelwert. Dieser Wert ist dann entscheidend, wenn es um die Beurteilung der Sequenz- und Abbaulängen von einzelnen Peptiden geht. Die folgende Tabelle betrachtet für die N-terminale Spaltung die Summe der Sequenzinformation verschiedener Peptide, wie sie aus Sequenzierungsexperimenten mit APM, AAP und API für diese Peptide resultieren (Mittelwert). Auf der C-terminalen Seite wurde die Summe der Sequenzinformation aus den Sequenzierungsexperimenten mit CPY, CPP und sb(CP-I) berechnet. Auch bei der Summe aus N- und C-terminaler Sequenzierung in Tabelle 28 sind Überlappungen der erhaltenen N- und C-terminalen Teilsequenzen zu berücksichtigt (vergleichbar P18-2107 in Tabelle 27, siehe oben).

Tabelle 28: Statistik der N- und C-terminalen Sequenzierungen einiger synthetischer Peptide – Zusammenfassung

Abbau	Sequenzabdeckung		Mittelwert Abbaulänge	Mittelwert Sequenzlänge
	Abbau	Sequenz		
N-terminal				
N-terminal	54%	54%	6-7	6-7
C-terminal				
C-terminal	42%	38%	7-8	6-7
C- & N-terminal				
Summe C- und N-terminal	56%	53%	11	10

Betrachtet man für die N-terminale Sequenzierung die kombinierten Resultate von bis zu drei verschiedenen Sequenzierungsexperimenten (APM, AAP und API) eines Peptids, so nimmt also die erhaltene Sequenzlänge um 2-3 Aminosäuren gegenüber den Einzelexperimenten zu. Dies entspricht einer Zunahme der Sequenzinformation im Falle der kombinierten Resultate um 50-75% gegenüber den Einzelexperimenten. Für die C-terminale Leitersequenzierung – also die kombinierten Resultate der Sequenzierung mit CPY, CPP und CPW – errechnen sich die gleichen Werte. Damit nimmt die Sequenzabdeckung bezüglich der Sequenzlänge für die kombinierten Resultate gegenüber den Einzelexperimenten um maximal 36% bei N-terminaler und 19% bei C-terminaler Sequenzierung zu. Der errechnete Wert für die Summe aus C- und N-terminaler Spaltung in Tabelle 28 bezieht sich auf diejenigen Peptide in Tabelle 26, die sowohl N- als auch C-terminal sequenziert wurden.

Die geringe Differenz der Sequenzabdeckungswerte für „Abbau“ und „Sequenz“ in Tabelle 27 und Tabelle 28 (Mittelwert 3%; maximal 7%), ist zudem ein deutliches Zeichen, dass Sequenzlücken in den Leiterspektren nur selten zu beobachten sind. Demnach ist die Kinetik des Aminosäuresäureabbaus mit den meisten der angewendeten Exopeptidasen bereits als weitgehend optimal zu beurteilen.

3.1.3 Massengenauigkeit

Bei den Sequenzierungen der synthetischen Peptide wurde auch untersucht, wie groß die üblicherweise zu erwartenden Abweichungen Δm_{AS} gegeben durch die Differenz der im Leiterspektrum gefundenen experimentellen Aminosäuremasse $m_{AS(exp)}$ von den tatsächlichen Aminosäuremassen $m_{AS(mono)}$ sind. Bezogen wurde dabei auf monoisotopische Massen. Bei

122 sequenzierten Aminosäuren betrug der Mittelwert dieser Abweichung im untersuchten Massenbereich (37 ± 23) mDa. Dies entspricht ca. (34 ± 21) ppm bis (13 ± 8) ppm. Dieser Wert ist unabhängig davon, ob eine vorherige Kalibrierung des Spektrums durchgeführt wurde oder nicht. Für die Massendifferenzen im Spektrum spielt eine Kalibrierung also wie erwartet keine Rolle. Der erhaltene Wert für Δm_{AS} zeigt, dass die MALDI-MS als Detektionsmethode für die Leiterpeptide mit den gegebenen Geräte- und Messparametern zur Unterscheidung nahezu aller 20 natürlichen Aminosäuren geeignet ist. Während für das Paar Leucin / Isoleucin ohnehin keine Unterscheidung auf massenspektrometrischem Weg zu erwarten ist, ist mit diesem Wert für Δm_{AS} jedoch auch die erhoffte Auflösung der Aminosäuren Glutamin und Lysin, deren Massendifferenz 36,38 mDa beträgt, auf direktem Weg nicht möglich. Für diese Unterscheidung erscheint eine selektive Derivatisierung einer der beiden Aminosäuren vor der Sequenzierung als zwingend notwendig.

3.1.4 Empfindlichkeit und Dynamik

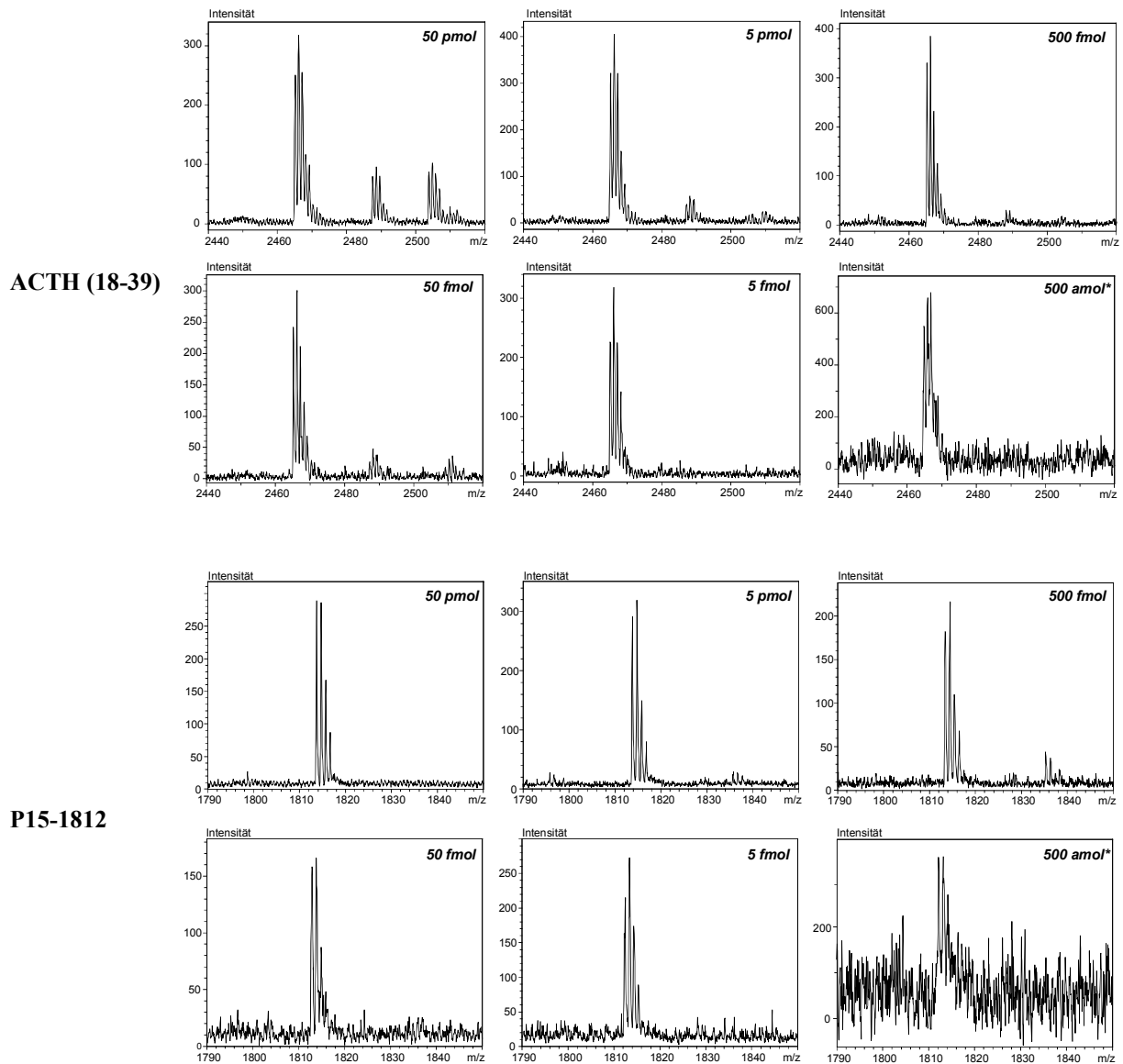
Untersuchungen über Empfindlichkeit und Dynamik der enzymatischen Peptidsequenzierung in Verbindung mit UV-MALDI-TOF-MS wurden anhand der C- und N-terminalen Sequenzierung der folgenden vier Peptide durchgeführt:

- ❖ ACTH (18-39); Sequenz: RPVKVYPNGAEDESAEAFPLEF; $[M+H]^+ = 2465,1989$ Da
- ❖ P15-1812; Sequenz: IKVTLVFEHVDQDLR; $[M+H]^+ = 1812,0020$ Da
- ❖ P12-1303; Sequenz: AGHEYGAEALER; $[M+H]^+ = 1302,6078$ Da
- ❖ P10-1213; Sequenz: FLDTLVVLHR; $[M+H]^+ = 1212,7105$ Da

Die zu sequenzierenden Peptidmengen wurden vom mittleren Pikomol-Bereich bis in den oberen Attomol-Bereich variiert.

Zunächst wurden die UV-MALDI-MS Spektren der Ausgangspeptide im Stoffmengenbereich von 50 pmol bis 500 amol untersucht. Durch diese Messungen sollte festgestellt werden, ob nach der Signalintensität des reinen Ausgangspeptids für dessen vorgelegte Stoffmenge überhaupt mit einer erfolgreichen Leitersequenzierung zu rechnen ist. Die Präparation der Proben erfolgte hierzu, in Analogie zur späteren Präparation bei der Leitersequenzierung, *on-target* mit DHB-Matrix wie unter Punkt 2.4 beschrieben. Abbildung 22 und Abbildung 23 zeigen die Resultate der Messungen.

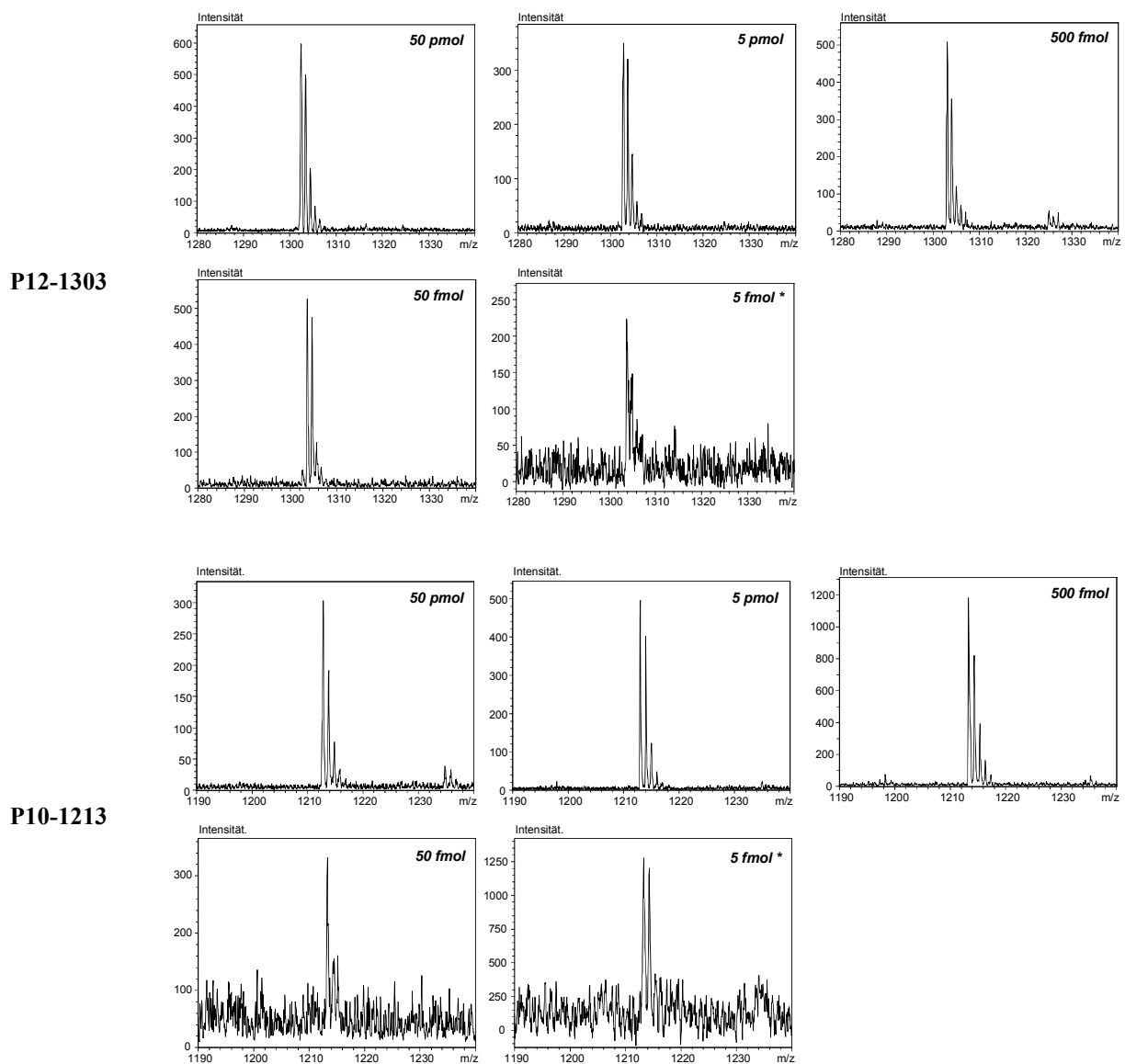
Abbildung 22: Signalintensitäten der Peptide ACTH (18-39) und P15-1812 im Bereich von 50pmol bis 500amol; Präparation *on-target* mit DHB; 10 Laserschüsse/Spektrum (Spektren mit * neben der Stoffmengenangabe: 100 Laserschüsse/Spektrum)



Die Spektren in Abbildung 22 zeigen für ACTH (18-39) und P15-1812 sehr gute Signalintensitäten im Bereich von 50 pmol bis 5 fmol. Die absoluten Intensitäten sind hier nicht direkt vergleichbar, denn die Abschwächung des UV-Lasers wurde an die jeweils gemessene Peptidmenge im Hinblick auf die Auflösung bei ausreichender Signalintensität optimiert. Die Peptidmengen erscheinen mit Blick auf die resultierenden Signalintensitäten jedoch in diesem Bereich für eine erfolgreiche Sequenzierung generell geeignet. Bei der Messung von 500 amol der beiden Peptide lassen sich befriedigende Signalintensitäten erst dann erhalten, wenn die Zahl der akkumulierten Spektren stark erhöht wird (100 summierte

Einzelspektren bei der Messung von 500amol Peptid, im Vergleich zu 10 summierten Einzelspektren bei der Messung von 50pmol – 5fmol Peptid). Die schlechten Signal/Rausch-Verhältnisse bei 500amol lassen vermuten, dass eine Sequenzierung dieser Peptidmengen keine zufriedenstellenden Resultate liefern wird, da die erhaltenen Leiterpeptide ebenfalls nur sehr geringe Intensitäten besitzen werden. Analog lassen sich die Spektren der Peptide P12-1303 und P10-1213 in Abbildung 23 interpretieren, allerdings wird hier bereits mit 50fmol eine für die Sequenzierung kritische Menge erreicht sein.

Abbildung 23: Signalintensitäten der Peptide P12-1303 und P10-1213 im Bereich von 50pmol bis 5fmol; Präparation *on-target* mit DHB; 10 Laserschüsse/Spektrum (Spektren mit * neben der Stoffmengenangabe: 100 Laserschüsse/Spektrum)



Die Abbildungen auf den folgenden beiden Seiten zeigen die Resultate der C-terminalen *on-target* Sequenzierung der Peptide ACTH (18-39), P15-1802 und P12-1303. Abbildung 24 zeigt für ACTH (18-39), dass eine enzymatische Sequenzierung in Kombination mit UV-MALDI-TOF-MS sehr gut Resultate bis zu einer ACTH-Konzentration von 10fmol erbringt. Die maximale Sequenzinformation von 15 Aminosäuren (NGAEDESAEAFPLEF), die sich aus einer Sequenzierung von 50pmol ergibt, bleibt bis hinunter zu einer sequenzierten ACTH-Stoffmenge von 100fmol vollständig erhalten. Bereits unterhalb einer sequenzierten Stoffmenge von 500fmol ACTH nimmt das Signal/Rausch-Verhältnis deutlich sichtbar ab. Ab einer sequenzierten Stoffmenge von 50fmol nimmt der Informationsgehalt des Leiterspektrums ebenfalls ab. Die Leiterpeptide 16 (RPVKVYP), 15 (RPVKVYPN) und 13 (RPVKVYPNGA) treten nicht mehr in Erscheinung. Dementsprechend beträgt die direkte Sequenzinformation noch 11 Aminosäuren (DESAEAFPLEF). Die Lücke zwischen Leiterpeptid 12 und 14 mit einer Massendifferenz von ca. 200 Da gibt keinen eindeutigen Hinweis auf die Aminosäuren Ala und Glu, denn theoretisch trifft diese Massendifferenz auch auf das Paar Pro / Cys zu. Die Reihenfolge der abgespaltenen Aminosäuren ist ebenfalls unbestimmt. Sequenzierte ACTH-Mengen unter 10fmol zeigen im wesentlichen dann nur noch die bereits bei größeren Stoffmengen dominierenden Signale der Leiterpeptide 5 (RPVKVYPNGAEDESAEA) und 6 (RPVKVYPNGAEDESAE) und somit nur die Abspaltung eines Alanins (Spektren nicht gezeigt).

Die C-terminalen MALDI-MS Leiterspektren der Sequenzierungen von 5pmol bis 50fmol der Peptide P15-1812 und P12-1303 zeigt Abbildung 25. Die Leiterspektren für 50pmol sind hier nicht mehr aufgeführt. Sie zeigen – analog zur Situation bei ACTH – die gleiche Information wie die Leiterspektren bei 5pmol sequenzierter Peptidmenge. Für P15-1812 (linke Spalte in Abbildung 25) ist die erhaltene Sequenzinformation im Bereich von 50pmol bis 50fmol nahezu unverändert. Bei Sequenzierung von 5fmol geht hier jedoch die Sequenzformation vollständig verloren. Nur das intensivste Leiterpeptid 3 (IKVTLVFEHVDQD) tritt hier noch auf. Bei P12-1303 (rechte Spalte in Abbildung 25) treten vergleichbare Probleme bereits bei der Sequenzierung von 50fmol auf. Die Massendifferenz von ca. 242 Da zwischen Leiterpeptid 2 und 4, die der Abspaltung von [LE] entspricht, trifft auf eine Vielzahl weiterer Aminosäurekombinationen zu ([NQ], [NK], [GAN], [GGQ] oder [GGGA]).

Abbildung 24: C-terminale *on-target* Sequenzierung von 50pmol bis 10fmol ACTH (18-39) mit CPY; Ein Stern (*) über einem Signal zeigt dessen Zugehörigkeit zur untersuchten Peptidleiter.

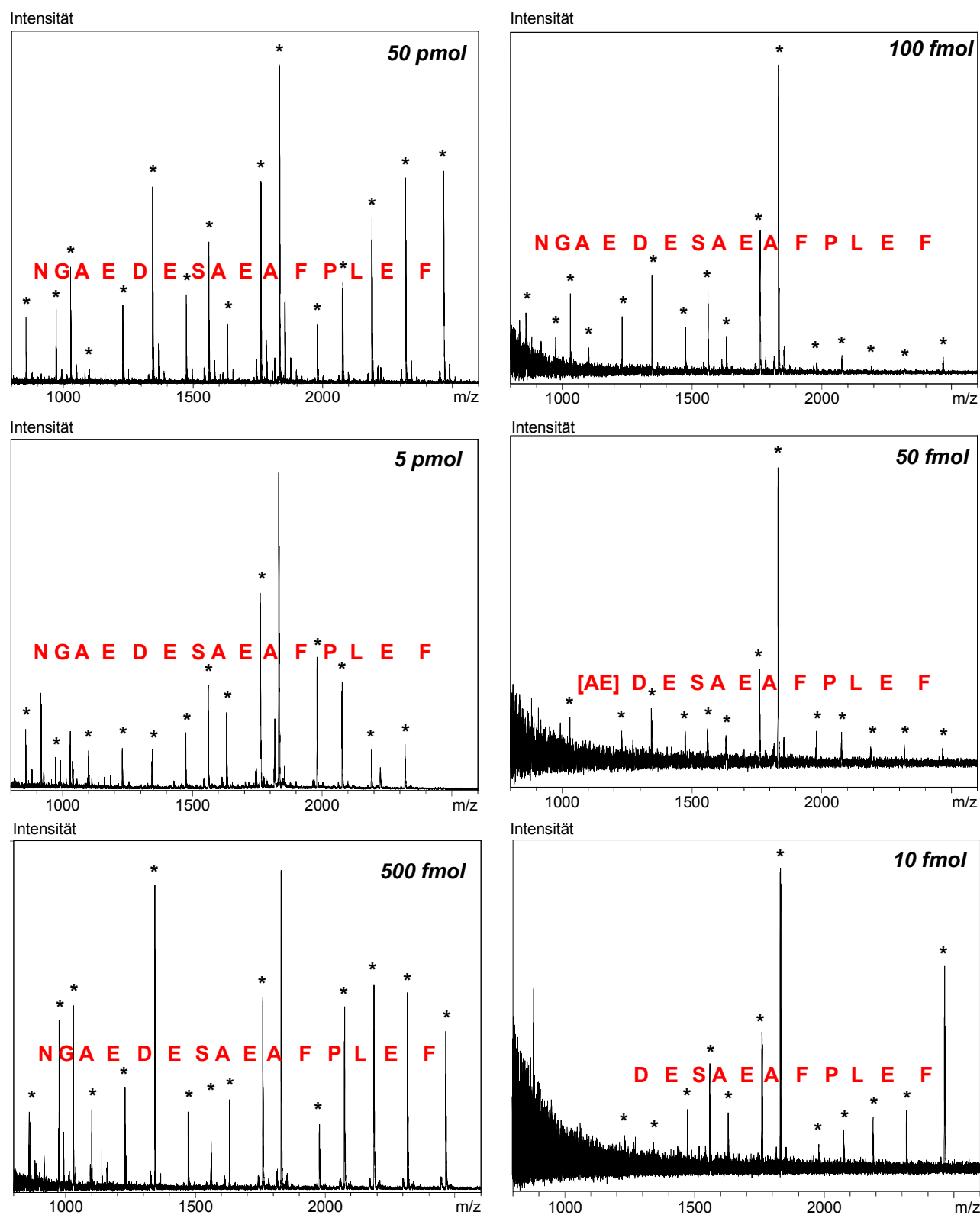
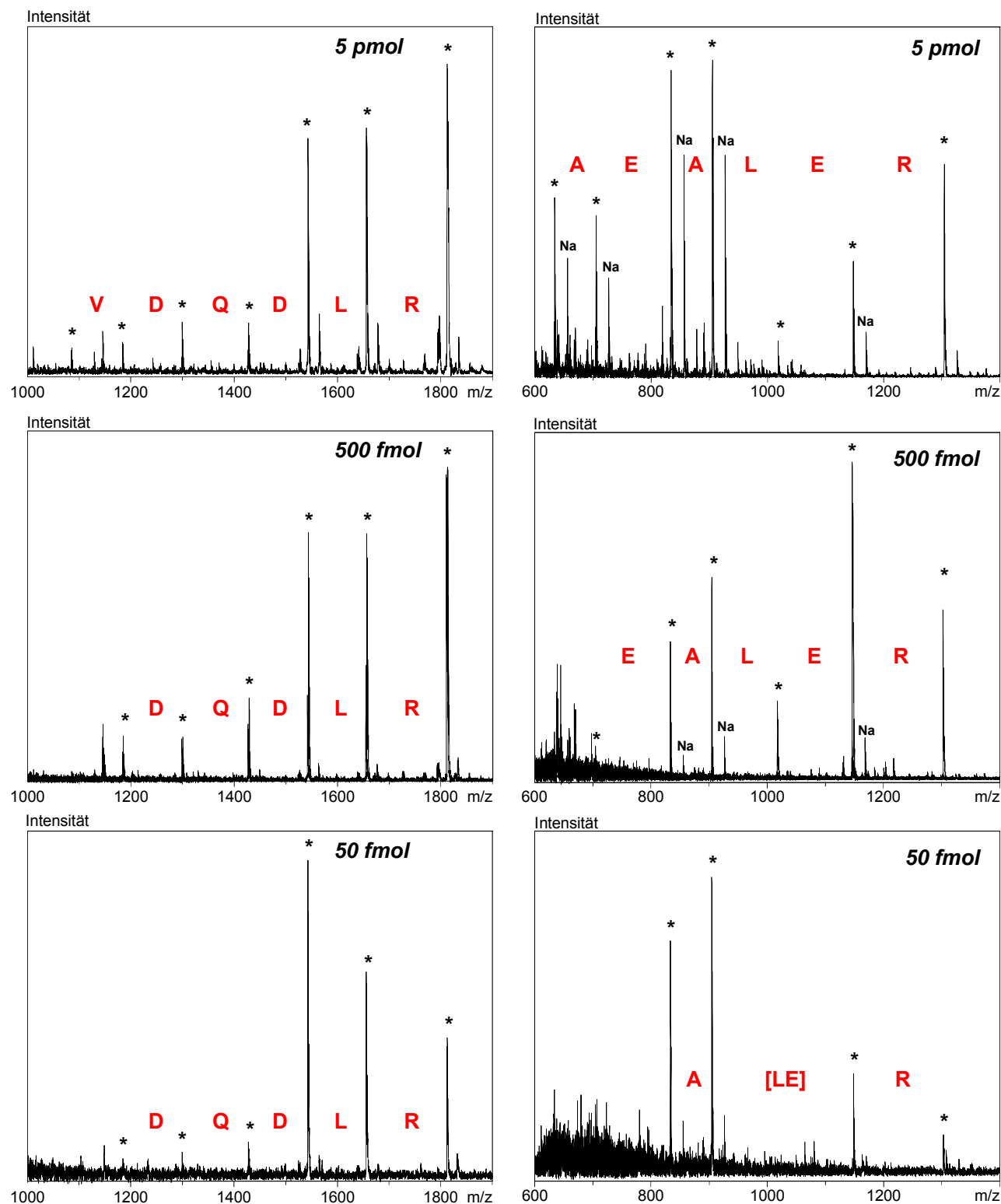


Abbildung 25: C-terminale *on-target* Sequenzierung von 5pmol bis 50fmol P15-1812 (linke Spalte) und P12-1303 (rechte Spalte) mit sb(CP-II) (CPB+CPY); Ein Stern (*) über einem Signal zeigt dessen Zugehörigkeit zur untersuchten Peptidleiter.



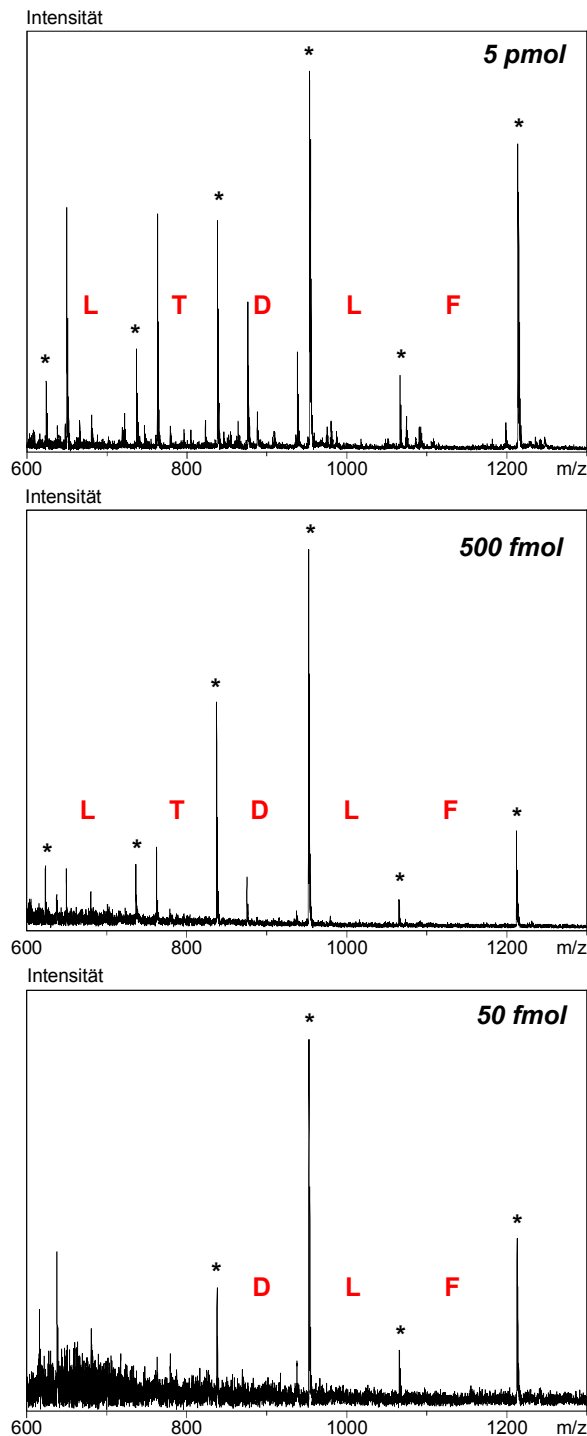
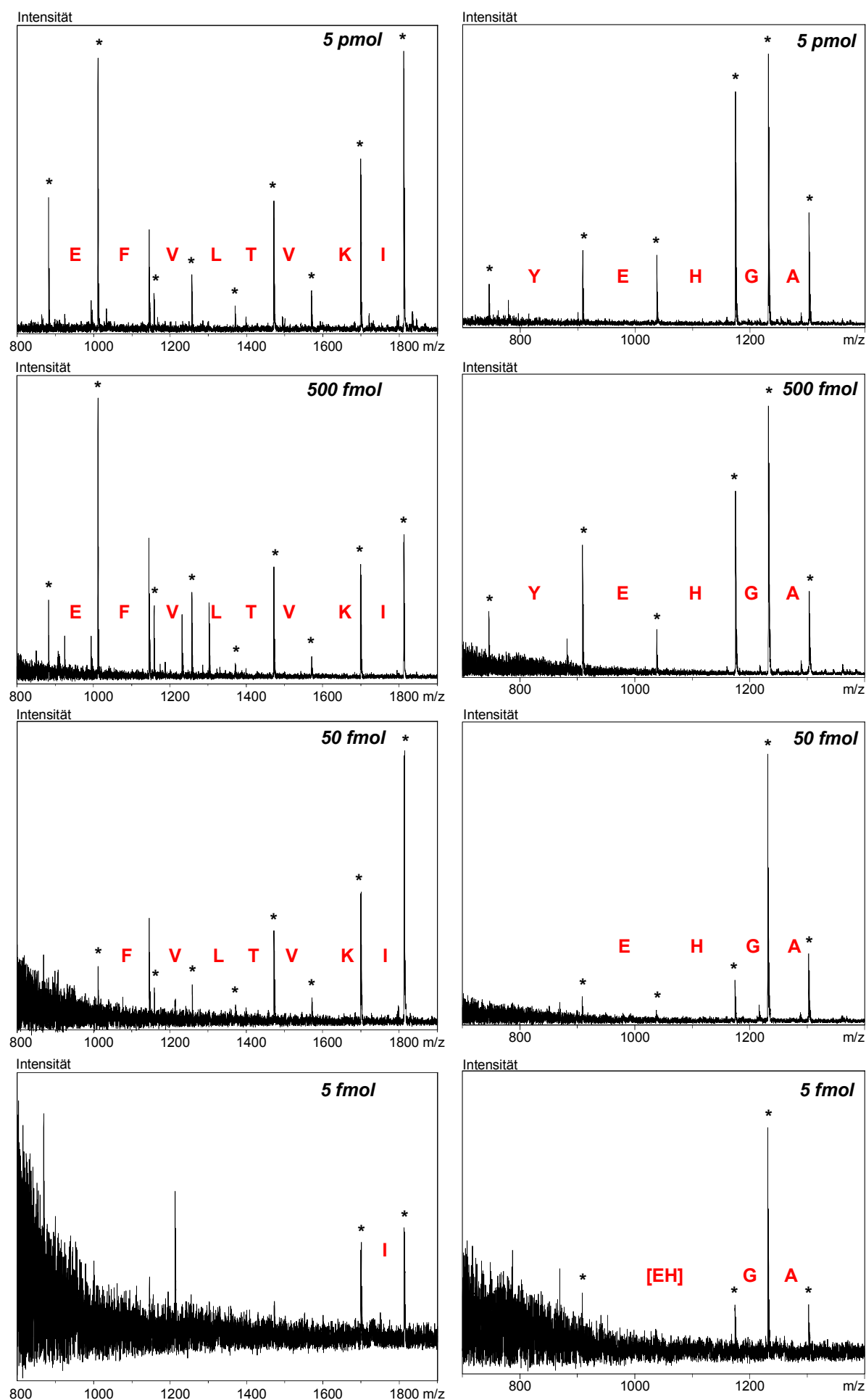


Abbildung 26 (oben): N-terminale *on-target* Sequenzierung von 5pmol bis 50fmol P10-1213 mit APM₁; Ein Stern (*) über einem Signal zeigt dessen Zugehörigkeit zur untersuchten Peptidleiter.

Für die N-terminale Sequenzierung sind Leiterspektren für verschiedene Peptide in Abbildung 27 und Abbildung 26 dargestellt. Die Ausgangskonzentrationen des sequenzierten Peptids wurden dabei im Bereich zwischen 50pmol und 5fmol variiert. Die APM Sequenzierungen der synthetischen Peptide P15-1812 und P12-1303 in Abbildung 27 (nächste Seite) zeigen zwischen 50pmol und 50fmol nahezu vergleichbare Resultate bezüglich der erhaltenen Sequenzinformation. (Die Resultate für 50pmol sind denen für 5pmol identisch und sind deshalb in Abbildung 27 nicht dargestellt). Im unteren Femtomol-Bereich (≤ 5 fmol) ist die aus Leiterspektren gewonnene Sequenzinformation im Gegensatz zum mittleren Femtomol-Bereich (≥ 50 fmol) drastisch reduziert. Für P10-1213 nimmt die Sequenzinformation bereits ab 50fmol sequenzierten Peptids um 40% gegenüber der Sequenzierung von 500fmol ab. Das MALDI-MS einer Leitersequenzierung von 5fmol des P10-1213 zeigt keine Leiterpeptidsignale mehr.

Abbildung 27 (folgende Seite): N-terminale *on-target* Sequenzierung von 5pmol bis 5fmol P15-1812 (linke Spalte) und P12-1303 (rechte Spalte) mit APM₁; Ein Stern (*) über einem Signal zeigt dessen Zugehörigkeit zur untersuchten Peptidleiter.



3.1.5 pH-Bedingungen

Für die kombinierte Anwendung mehrerer Peptidasen, insbesondere für die Anwendung von Peptidasen in einer Mischung, müssen Pufferbedingungen existieren, bei denen alle eingesetzten Peptidasen für die Sequenzierung optimal arbeiten. Hier ist neben der richtigen Wahl des verwendeten Puffersystems und dessen Ionenstärke vor der richtige pH-Wert entscheidend. Es muss in erster Linie ein pH-Bereich gefunden werden, innerhalb dessen alle kombinierten Enzyme noch eine für die Sequenzierung ausreichende Aktivität besitzen.

Die Werte für die pH-Optima der verwendeten Peptidasen in Tabelle 2 und Tabelle 3 (Seite 16 ff.), sowie Tabelle 10 und Tabelle 12 (Seite 67 ff.) zeigen, dass vor allem für den Einsatz verschiedener Carboxypeptidasen in Gemischen ein Kompromiss für den angewendeten pH-Bereich erforderlich ist. Während sich die Intervalle der pH-Optima der eingesetzten Amino-peptidasen sehr stark überlappen (um pH 8,0), reichen die pH-Optima der eingesetzten Carboxypeptidasen vom sauren (z.B. pH 4,0 für CPW) bis in den basischen Bereich (z.B. pH 7,0-9,0 für CPB). Anhand der folgenden Sequenzierungen mit den Carboxypeptidasen CPB, CPY, CPP und CPW sollte festgestellt werden, ob ein geeigneter pH-Bereich für die Sequenzierung mit beliebigen Carboxypeptidase-Kombinationen überhaupt existiert. Sequenziert wurden die Peptide:

- ❖ P15-1812; Sequenz: IKVTLVFEHVDQDLR; $[M+H]^+ = 1812,0020$ Da
- ❖ P12-1303; Sequenz: AGHEYGAEALER; $[M+H]^+ = 1302,6078$ Da
- ❖ P10-1213; Sequenz: FLDTLVVLHR; $[M+H]^+ = 1212,7105$ Da
- ❖ Neurotensin; Sequenz: (pyro)ELYENKPRRP; $[M+H]^+ = 1672,9176$ Da

Die Grafiken in Abbildung 28 und Abbildung 29 zeigen die erhaltenen Resultate. Der für die jeweilige Carboxypeptidase geeignete pH-Bereich für die Leitersequenzierung ist grau unterlegt. Ein Vergleich aller Grafiken zeigt, dass der noch zusätzlich für alle Carboxypeptidasen dunkelgrau hervorgehobene Bereich zwischen pH 5,5 und 6,5 für beliebige Carboxypeptidase-Kombinationen einen guten Kompromiss darstellt. Bei der Anwendung aller Einzelpeptidasen ergibt sich für die Sequenzierung in diesem pH-Bereich ein Resultat, dass auch der maximal möglichen Sequenzinformation für die Peptidase entspricht. Als geeignetes Puffersystem, erweist sich für diesen pH-Bereich ein Citratpuffer hergestellt aus Natriumcitrat oder besser Ammoniumcitrat. Diese Puffersysteme sind auch kompatibel zur MALDI-MS Messung. Die Tendenz zur Bildung von Natriumaddukten bei Verwendung von Natriumcitrat führt im gewissen Ausmaß zur einer Signalunterdrückung der Leiterpeptide (Suppressionseffekt) und damit zu einer geringeren Empfindlichkeit (siehe unten; 3.5 Einfluss der MALDI-Matrix und Suppressionseffekte).

Legende:

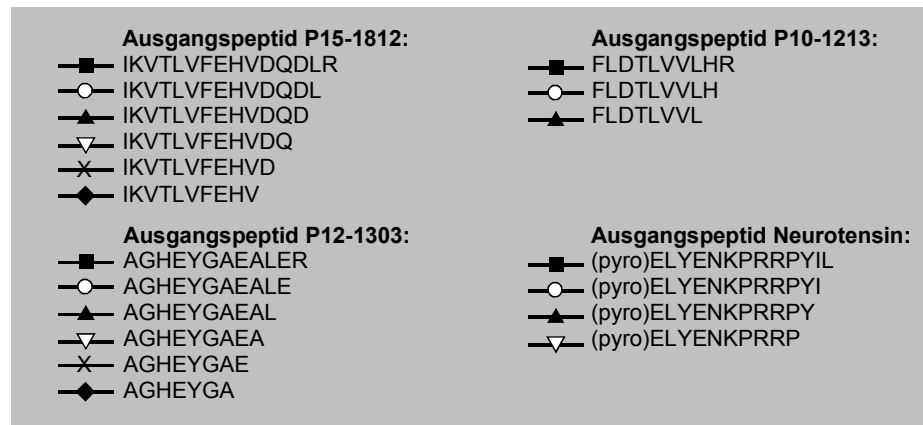


Abbildung 28: Intensitätsprofile der Leiterspektren von drei synthetischen Peptiden bei der Sequenzierung mit CPP, CPW. Alle Intensitäten wurden auf das höchste Leiterpeptid im jeweiligen Spektrum normiert.

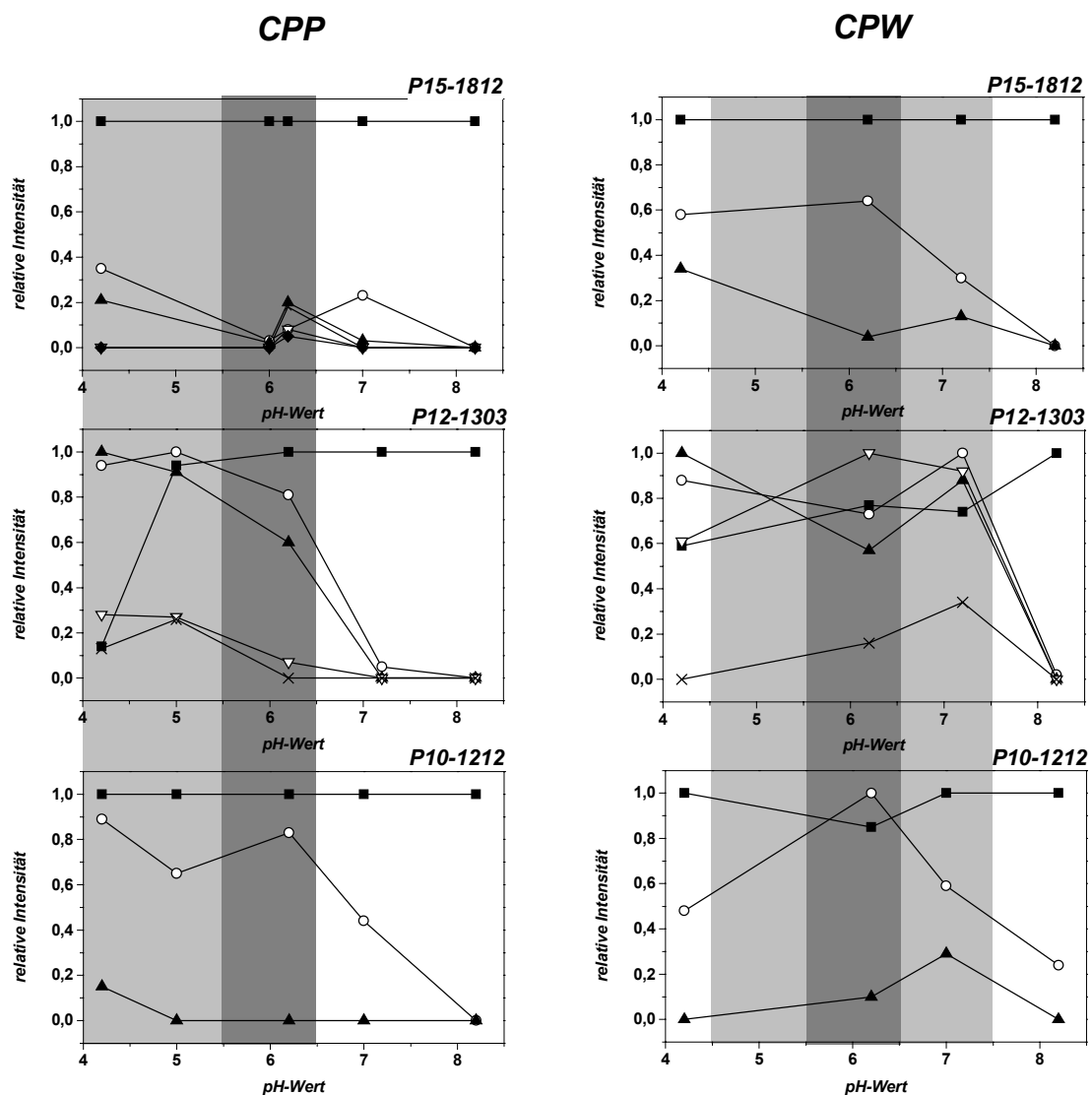
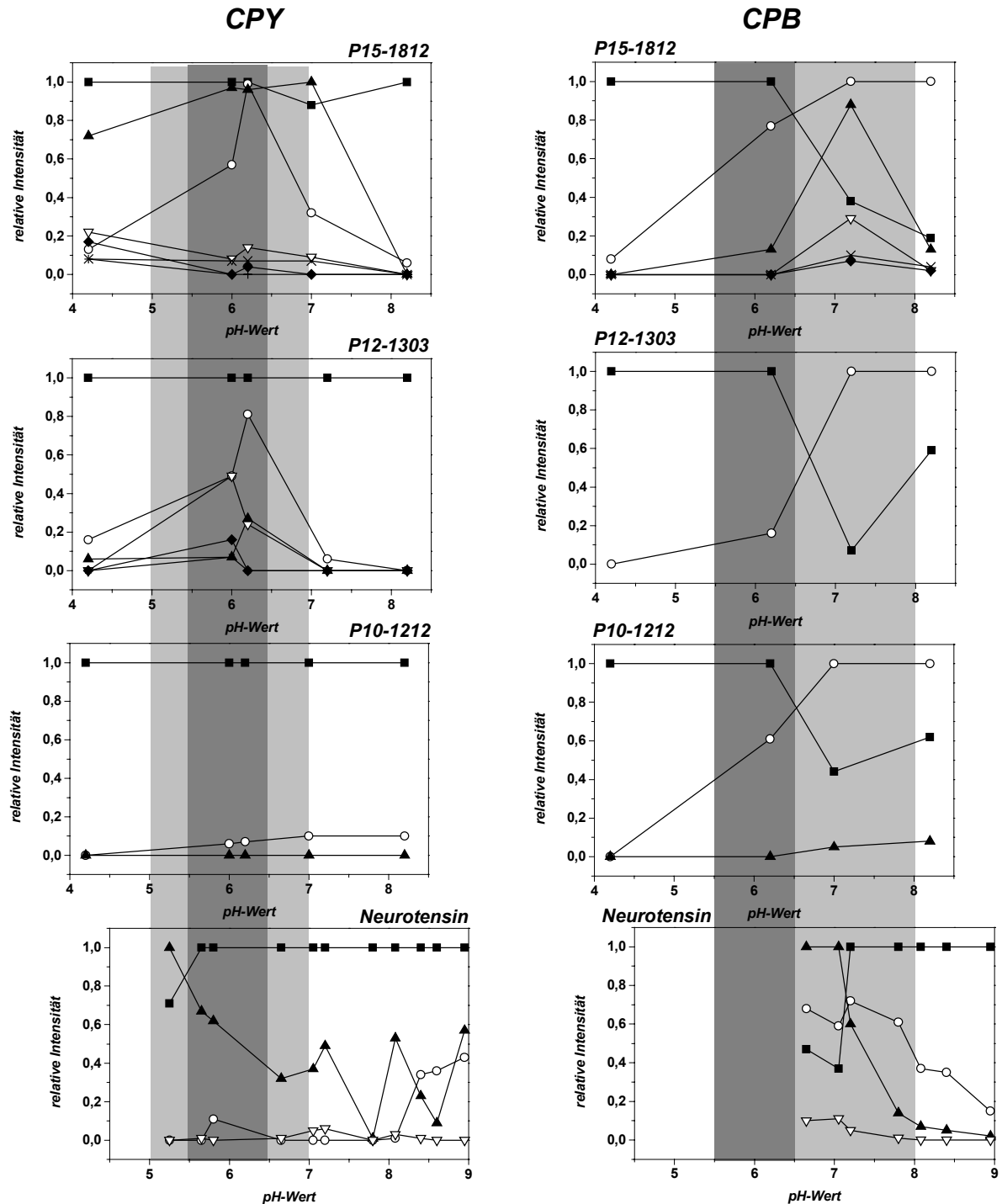


Abbildung 29: Intensitätsprofile der Leiterspektren von vier synthetischen Peptiden bei der Sequenzierung mit CPY, CPB. Alle Intensitäten wurden auf das höchste Leiterpeptid im jeweiligen Spektrum normiert.



Weitere Bedingungen bei den Sequenzierungen:

Spaltungstemperatur 27°C; Puffersysteme: 2,5mM Natriumcitrat-HCl bei pH < 7 und 5mM Tris-HCl bei pH > 7;

3.2 Proteolytische Spaltpeptide

Die nachfolgenden Tabellen enthalten alle proteolytischen Spaltfragmente der untersuchten Proteine, die nach der HPLC-Aufreinigung der Spaltgemische wiedergefunden und letztlich auch einer enzymatischen Sequenzierung unterzogen wurden. Sofern nicht die Sequenzierung eine direkte Zuordnung eines bestimmten Spaltfragments zu einem definierten Sequenzabschnitt des untersuchten Proteins ermöglichte, konnte in den allen Fällen eine indirekte Zuordnung über die Fragmentmasse erfolgen. Die Proteinsequenzen, die hierbei zum Vergleich mit den ermittelten Massen und Sequenzabschnitten herangezogen wurden, wurden bis auf die Sequenz des *DrTI* (*Serin Proteinase Inhibitor*) der SWISS-PROT Datenbank entnommen. Dabei wurden auch mögliche Varianten, Konflikte und post-translationale Modifikationen, die in der Datenbank zu den einzelnen Proteinen aufgeführt sind, berücksichtigt. Die nachfolgende Aufstellung gibt einen Überblick über die Quellen der zur Auswertung herangezogenen Proteinsequenzen:

Tabelle 29: SWISS-PROT Einträge der sequenzierten Proteine

Trivialname	SWISS-PROT		
	Eingetragene Datenbank-Bezeichnung	Accession No.	Release
β -Lactoglobulin	beta-Lactoglobulin Precursor (beta-LG) (Allergen Bos D5) – Bos taurus (bovine)	P02754	40, Oktober 2000
β -Casein	beta-Casein Precursor – Bos taurus (bovine)	P02666	40, Oktober 2000
α -Casein	alpha-S1 Casein Precursor – Bos taurus (bovine)	P02662	38, Juli 1999
Myoglobin	Myoglobin – Equus caballus (horse)	P02188	40, Oktober 2000
Cytochrom C	Cytochrom C – Equus caballus (horse)	P00004	38, Juli 1999
Aldolase	Fructose-Biphosphate Aldolase A (EC 4.1.2.13) (Muscle-Type Aldolase) – Oryctolagus cuniculus	P00883	38, Juli 1999
Alkoholdehydrogenase	Alcohol Dehydrogenase I (EC 1.1.1.1) – Saccharomyces Cervisiae (baker's yeast)	P00330	35, November 1997
Rinderserum Albumin	Serum Albumin Precursor – Bos taurus (bovine)	P02769	37, Dezember 1998
Schweineserum Albumin	Serum Albumin Precursor (Fragment) – Sus scrofa (pig)	P08833	36, Juli 1998

Die angegebene Aminosäuresequenz des *Serin Protease Inhibitors DrTI* wurde im Frühjahr 2000 am MPI für Biochemie von Herrn Reinhard Mentele (Abteilung für klinische Chemie und Biochemie der Ludwig-Maximilians-Universität München) über Edman-Sequenzierung

ermittelt und freundlicherweise zum Vergleich zur Verfügung gestellt. Diese Sequenz ist noch nicht als Eintrag in einer Proteindatenbank enthalten.

3.2.1 Sequenzen

Die in den folgenden Tabellen angegebenen Peptidmassen bezeichnen die monoisotopischen Massen der einfach positiv geladenen Peptidsplaltfragmente. Die Spalte „Abschnitt“ bezeichnet die Position des Sequenzabschnitts im Protein. (Falls für das jeweilige Protein zutreffend wurden auch die Positionen der Signalsequenz im Precursor miteinbezogen.) Mit „MC“ (*missed cleavage*) wird die Zahl der durch die Endopeptidase ausgelassenen Spaltungsstellen in einem proteolytischen Spaltfragment bezeichnet. Des weiteren kommen folgende Bedeutungen den verschiedenen Schreibweisen und Symbolen bei einzelnen Sequenzen zu:

Tabelle 30: Schreibweisen und Symbolerklärungen zu den Sequenztabelle der proteolytischen Spaltfragmente

unterstrichener Sequenzteil ohne Klammerung	z.B. <u>X</u> : durch enzymatische Abspaltung abgebauter und lesbarer Sequenzteil aus dem Leiterspektrum
unterstrichener Sequenzteil mit Klammerung	z.B. [<u>X</u>]: durch enzymatische Abspaltung abgebauter, aber nicht lesbarer Sequenzteil aus dem Leiterspektrum: Lücke im Leiterspektrum
doppelte Unterstreichung	z.B. <u><u>X</u></u> : Sequenzteil wird sowohl N- als auch C-terminal abgebaut
fett gedruckter, unterstrichener Sequenzteil	z.B. <u>X</u> rechnerisch ermittelte Aminosäure durch die Summe der Information aus C- und N-terminaler Teilsequenz
vertikaler Strich () in der Sequenz	Trennt die Information aus C- und N-terminaler Teilsequenz, falls diese direkt aneinander grenzen
PE: ###	Peptidfragment enthält pyridylethyliertes Cystein durch die Behandlung mit Vinylpyridin; mit Angabe der Position (###) im Protein
FS: AS###	Fehlsplaltung der Endopeptidase, d.h. Spaltung an einer Aminosäure, die nicht der Spezifität der Endopeptidase entspricht mit Angabe von Aminosäure (AS) und Position (###) im Protein
MSO: ###	Methionin an Position ### liegt oxidiert als Methioninsulfoxid vor
Phos: ###	Phosphorylierte Aminosäure an Position ### im Protein
Carb: ###	Glycosylierte Aminosäure an Position ### im Protein
###: X->Y	Austausch einer Aminosäure Y für X gegenüber der angegebenen Standard-SWISS-PROT Sequenz mit Angabe der Position (###) im Protein

3.2.1.1 Sequenzen aus Spaltungen mit Endoproteinase LysC

Masse	Abschnitt	Sequenz	MC
<i>Cytochrom C (P00004)</i>			
604,345	56- 60	GITWK	0
634,392	9- 13	IFVQK	0
678,382	74- 79	YIPGTK	0
779,448	80- 86	MIFAGIK	0
806,477	73- 79	KYIPGTK	0
1296,717	28- 39	TGPNLHGLFGRK	0
1350,726	89- 99	TEREDLIAYLK	0
1470,686	40- 53	TGQAPGFTYTDANK	0
1478,821	89-100	TEREDLIAYLKK	1
1495,698	61- 72	EETLMEYLENPK	0
1633,819	9- 22	IFVQKCAQCHTVEK	1
2081,026	56- 72	GITWKEETLMEYLENPK	0
<i>Myoglobin (P02188)</i>			
1360,758	134-145	ALELFRNDIAAK	0
1378,842	64-77	HGTVVLTAALGGILK	0
1502,669	119-133	HPGDFGADAQGAMTK	0
1815,902	1- 16	GLSDGEWQQVLNVWGK	0
1885,022	103-118	YLEFISDAI IHV [LHS] K	0
2859,500	17- 42	VEADIAGHGQEVLI RLFTGHPETLEK	0
<i>DrTI (Serin Proteinase Inhibitor)</i>			
751,435	128-134	LGSLAYK	0
811,454	135-140	LVFCPK PE: 183	0
949,423	176-184	PRSGSETES	0
1482,759	163-175	SSDDSPFRVVFVK	0
1892,856	141-157	SSS [GSC] SDIGINYEGR FS: R157 PE: 146	0
2143,094	105-124	VAIGGSEDHPQGELVRGFFK FS: R104	0
2279,101	99-120	DSGEARVAIGGSEDHPQGELVR FS: R120	0
2433,220	141-162	SSSGSCSDIG INYEGRRSLVLK PE: 146	0
2651,244	73- 94	IYTDTELEIEFVEKPDCAESSK PE: 89	0
2758,355	99-124	DSGEARVAIGGSEDHPQGELVRGFFK	0
3128,577	99-127	DSGEARVAIGGSED [HPQ] GELVRGFFKIEK	1
3431,764	6- 38	VYDIEGYPVFLGSEYYIVSAIIGAGGG [VR] PGR FS: R38	0
3601,780	41- 72	GSMCPMSIIQEQLQMGPLPVRFSPE [ESQ] GK FS: R40 PE: 44	0
3858,829	39- 72	TRGSMCPMSIIQEQLQMGPLPVRFSPE [ESQ] GK FS: R38 PE: 44	0

Masse	Abschnitt	Sequenz	MC
<i>β-Lactoglobulin (P02754)</i>			
573,361	87-91	IIAEK	0
674,424	94- 99	IPAVFK	0
916,437	100-107	IDALNENK	0
933,544	17- 24	LIVTQTMK FS: A16	0
1065,583	108-116	VLVLDTDYK	0
1169,504	77-85	WENGECAQK PE: 82	0
1571,879	94-107	IPAVFKIDALNENK	1
2580,863	158-178	ALPMHIRLSFNPTQLEEQCHI PE: 176	0
4188,775	118-151	YLLFCMENSAEPEQSLACQCLVRTPEVDDEALEK PE: 122, 135, 137	0
5001,617	31- 76	VAGTWYSLAMAASDISLLDASAPLRVYVEELKPTPEGDLI [LQK]	0
<i>Aldolase (P00833)</i>			
723,331	140-146	DGADFAK	0
763,471	208-214	VLAAY [YK]	0
909,513	147-152	WRCVLK PE: 149	0
937,463	322-329	AAQEEYVK	0
1131,562	200-207	RCQYVTEK PE: 201	0
1332,700	28- 41	GILAADESTGSIK	0
1336,715	330-341	RALANSLACQK PE: 338	0
1341,691	1- 12	PHSHPALTPQK	0
1342,711	87- 98	ADDGRPFQVIK	0
1469,786	1- 13	PHSHPALTPQKK	1
1505,138	230-242	PNMVTPGHACTQK MSO: 332	0
1505,843	14- 27	ELSDIAHRIVAPGK	0
1557,838	87-100	ADDGRPFQVIKSK	0
1709,920	215-229	ALSDHHIYLEGTLLK	0
1980,070	294-311	PWALTFSYGRALQASALK	0
2242,041	342-363	YTPSGQAGAAASESLFISNHAY	0
2639,453	289-311	CPLLKPWALTFSYGRALQASALK PE: 289	0
2970,463	111-138	GVVPLAGTNGETTTQGLDGLSERCAQYK PE: 134	0
3098,557	111-139	GVVPLAGTNGETTTQGLDGLSERCAQYKK PE: 134	0
4872,484	243-288	YSHEEIAMATVTALRRTPPA [VTGVTFSLGGQ] S [EEEASINLNAIN] K	0
5172,613	153-199	IGEHTPSALAIMENANVLARYASICQONGIVPIVEPEILPDGDHDLK PE: 177 MSO: 164	
5411,807	42- 86	RLQSIGTENTENRRFY [RQLLLTA] DDRVNPC [IG] GVILF [HE] TLY QK PE: 72	0

Masse	Abschnitt	Sequenz	MC
<i>Rinderserum Albumin (P02769)</i>			
818,425	562-568	<u>A</u> T <u>E</u> E <u>Q</u> L <u>K</u>	0
922,488	249-256	<u>A</u> E <u>F</u> <u>V</u> E <u>V</u> T <u>K</u>	0
935,375	581-587	<u>C</u> C <u>A</u> [<u>A</u> D <u>D</u>] <u>K</u> PE: 581, 582	0
974,458	37-44	<u>D</u> L <u>G</u> E <u>E</u> H <u>F</u> <u>K</u>	0
987,537	29-36	<u>S</u> E <u>I</u> A <u>H</u> R <u>F</u> <u>K</u>	0
1116,478	413-420	<u>Q</u> N <u>C</u> D <u>Q</u> <u>F</u> E <u>K</u> PE: 415	0
1120,546	310-318	<u>S</u> H <u>C</u> I <u>A</u> E <u>V</u> E <u>K</u> PE: 312	0
1142,714	548-557	<u>K</u> Q <u>T</u> A <u>L</u> V <u>E</u> L <u>L</u> <u>K</u>	1
1155,550	588-597	<u>E</u> A <u>C</u> F <u>A</u> V <u>E</u> G <u>P</u> <u>K</u> PE: 590	0
1163,631	66-75	<u>L</u> V <u>N</u> E <u>L</u> T <u>E</u> F <u>A</u> <u>K</u>	0
1305,716	402-412	<u>H</u> L <u>V</u> D <u>E</u> P <u>Q</u> N <u>L</u> I <u>K</u>	0
1387,675	300-309	[<u>E</u> C] <u>C</u> D <u>K</u> P <u>L</u> L <u>E</u> <u>K</u> PE: 301, 302	0
1415,687	569-580	<u>T</u> V <u>M</u> E <u>N</u> F <u>V</u> A <u>F</u> V <u>D</u> <u>K</u> MSO: 571	0
1467,587	89-100	<u>S</u> L <u>H</u> [<u>T</u> L] <u>F</u> G <u>D</u> E <u>L</u> <u>C</u> <u>K</u> PE: 99	0
1491,678	286-297	<u>Y</u> I <u>C</u> D <u>N</u> Q <u>D</u> T <u>I</u> S <u>S</u> <u>K</u> PE: 288	0
1559,662	76-88	<u>T</u> C <u>V</u> A <u>D</u> E <u>S</u> <u>H</u> A <u>G</u> C <u>E</u> <u>K</u> PE: 77, 86	0
1598,687	375-386	<u>E</u> Y <u>E</u> A <u>T</u> L <u>E</u> E <u>C</u> C <u>A</u> <u>K</u> PE: 383, 384	0
1602,689	387-399	<u>D</u> D <u>P</u> H <u>A</u> C <u>Y</u> S <u>T</u> V <u>F</u> D <u>K</u> PE: 392	0
1624,804	139-151	<u>L</u> K <u>P</u> D <u>P</u> N <u>T</u> L <u>C</u> D <u>E</u> F <u>K</u>	0
1721,806	118-130	<u>Q</u> E <u>P</u> E <u>R</u> N <u>E</u> C <u>F</u> L <u>S</u> H <u>K</u> PE: 125	0
1843,788	184-197	<u>Y</u> N <u>G</u> V <u>F</u> Q <u>E</u> C <u>C</u> Q <u>A</u> E <u>D</u> <u>K</u> PE: 191, 192	0
1955,960	319-336	<u>D</u> A <u>I</u> P <u>E</u> N <u>L</u> P <u>P</u> L <u>T</u> A <u>D</u> F <u>A</u> E <u>D</u> <u>K</u>	0
2028,102	421-437	<u>L</u> G <u>E</u> Y <u>G</u> F <u>Q</u> N <u>A</u> L <u>I</u> V <u>R</u> Y <u>T</u> R <u>K</u>	0
2116,913	101-117	<u>V</u> A <u>S</u> L <u>R</u> E <u>T</u> Y <u>G</u> D <u>M</u> A <u>D</u> C <u>C</u> E <u>K</u> PE: 114, 115 MSO: 111	0
2392,052	267-285	<u>E</u> C <u>C</u> H <u>G</u> D <u>L</u> L <u>E</u> C <u>A</u> D <u>D</u> R <u>A</u> D <u>L</u> A <u>K</u> PE: 268, 269, 276	0
2540,301	45-65	<u>G</u> L <u>V</u> L <u>I</u> A <u>F</u> S <u>Q</u> Y <u>L</u> Q <u>Q</u> C <u>P</u> F [<u>D</u> E] <u>H</u> V <u>K</u> PE: 58	0
2953,484	161-183	<u>Y</u> L <u>Y</u> E <u>I</u> A <u>R</u> R <u>H</u> P <u>Y</u> F <u>Y</u> A <u>P</u> E <u>L</u> L <u>Y</u> Y <u>A</u> N <u>K</u>	0
3144,510	499-523	<u>C</u> C <u>T</u> E <u>S</u> L <u>V</u> N <u>R</u> R <u>P</u> C <u>F</u> S <u>A</u> L <u>T</u> P <u>D</u> E <u>T</u> Y <u>V</u> P <u>K</u> PE: 499, 500, 510	0
3300,753	347-374	<u>D</u> A <u>F</u> L <u>G</u> S <u>F</u> L <u>Y</u> E <u>Y</u> S <u>R</u> R <u>H</u> P <u>E</u> Y <u>A</u> V <u>S</u> V <u>L</u> L <u>R</u> L <u>A</u> K	0

Masse	Abschnitt	Sequenz	MC
<i>β-Casein (P02666)</i>			
748,370	123-128	<u>EMP</u> FPK	0
780,498	185-191	VLPVPQK	0
1981,862	48- 63	FQSEEQQQ <u>T</u> EDELQDK	0
2061,828	48- 63	FQSEEQQQ <u>T</u> EDELQDK Phos: 50	0
3737,029	192-224	AVPYPQRDMPIQAFLLYQEPVL [GP] VRGPFPI [IV] MSO: 200	0
5316,853	64-112	IHPFAQTQSLVYPFPGPIPNSLPQNIPPLTQTPVVVPPF [LQP] E [VMG]] [VS] K Carb: 70, 72, 95	0
6359,256	129-184	YPVEPFTESQSLTLTDVENLHLPLPLLQSWMHQPHQPLPPTVMFPPQSV LSLSQSK Carb: 183	0

3.2.1.2 Sequenzen aus Spaltungen mit Endoproteinase GluC (Phosphatpuffer)

Masse	Abschnitt	Sequenz	MC
<i>α-Casein (P02662)</i>			
579,314	46- 50	VFGKE	0
664,333	72- 76	<u>I</u> KQME MSO: 75	0
763,365	71- 76	<u>D</u> IKQME	1
876,465	208-214	KT T MPLW	0
900,472	134-140	<u>RL</u> HSMKE	0
902,429	157-163	LAYFYPE	0
910,510	93- 99	<u>Q</u> KHIQ [KE]	0
913,478	105-111	<u>RY</u> <u>L</u> GYLE	0
938,387	126-133	IVPNSAEE Phos: 130	0
979,440	71- 78	DIKQMEAE	2
1049,563	46- 54	<u>V</u> FGKEKVNE	1
1134,560	205-214	NSGKT T MPLW 207: E->G	0
1229,631	164-172	<u>L</u> FRQFYQLD	0
1325,491	55- 65	<u>L</u> [SK]DIGSESTE Phos : 61, 63	2
1440,700	100-111	<u>DV</u> [PS]ERYLGYLE	2
1449,789	34- 45	NLLRFFVAPFPE	0
1664,934	16- 29	RPKHPI [KH] <u>Q</u> GLPQE	0
1756,084	112-125	<u>Q</u> LLRLKKYKVPQLE	0
1778,885	141-156	<u>GI</u> HAQQKEPMIGVNQE N-terminal : <u>GI</u> HAQQKE [PM] IGVNQE	0
2120,172	16- 33	RPKHPIKH <u>Q</u> GLPQEVLE	1
3463,600	173-204	AYPGAWYYVPLGTQYTDAPSFSDIPNPIGSE	

Masse	Abschnitt	Sequenz	MC
<i>Aldolase (P00833)</i>			
811,410	219-224	HHIYLE	0
881,400	190-197	<u>ILPDGDHD</u>	2
902,458	319-326	NLKAAQEE	0
953,457	82- 89	TLYQKADD	1
1051,521	355-363	SLFISNHAY	0
1069,535	156-165	HTPSALAIME	0
1085,537	1- 10	PHSHPALTPE	0
1116,562	133-140	<u>RCAQYKKD</u> PE: 134	0
1121,532	122-132	<u>TTTQGLGDLSE</u>	1
1244,646	198-206	<u>LKRCQYVTE</u> PE: 201	0
1245,680	307-318	<u>ASALKAWGGKKE</u> FS: Q306	0
1322,659	317-327	<u>KENLKAAQEEY</u> FS: K316, Y327	3
1359,647	133-143	<u>RCAQYKKDGAD</u> PE: 134	1
1394,665	340-354	<u>GKYTPSGQAGA [AAS] E</u> FS: 339	0
1547,839	35- 49	<u>STGSIAKRLQSIGTE</u>	0
1554,861	144-155	<u>FAKWRCV [LKI] GE</u> PE: 149	0
1598,828	1- 14	<u>PHSHPALTPEQKKE</u>	1
1658,883	68- 81	<u>RVNPCIGGVILFHE</u> PE: 72	0
1714,490	303-318	<u>RALQASALKAWGGKKE</u> FS: G302	0
1730,991	18- 34	<u>IAHRIVAPGKGILAADE</u>	0
1797,947	141-155	<u>GADFAKWRCV [LKI] GE</u> PE: 149	1
2021,015	35- 53	<u>STGSIAKRLQSIGTENTEE</u>	1
2046,134	15- 34	<u>LSDIAHRIVAPGKGILAADE</u>	1
2070,138	207-224	<u>KVLAAV [YK] ALS DHHIYLE</u>	1
2517,940	225-246	<u>GTLLKPNMVTTPGHACTQKYSHE</u> MSO: 164	0
2646,280	225-247	<u>G [TLL] KPNMVTTPGHACTQKYSHEE</u> PE: 239	0
2703,392	166-189	<u>NA [NVLARY] ASICQNGIVPIVEPE</u> PE: 177 N-terminal: <u>NA [NVLARYA] [SIC] QQN [GI] VPIVEPE</u>	0
2742,330	279-302	<u>ASINLNAINKCPL [LKP] WA [LTFSYG]</u> FS: G302	1
2917,462	327-354	<u>YVK [RAL] ANSLACQGKYTPSGQAGAAASE</u> PE: 338 N-terminal: <u>YVK [RAL] AN [SL] [AC] Q [GK] YTPSGQAGAAASE</u>	0

Masse	Abschnitt	Sequenz	MC
<i>Aldolase (P00833) – Fortsetzung</i>			
3016,662	82-109	TLYQKADDGRFPQVIKSKGGVVG [IK] VD	2
3203,826	90-121	GRFPQVIKSKGGVVG [VDKG] VVP [LAG] TNGE	1
3204,657	248-278	IAMATVTALRRTPPAVTGVTFLSGGQSEEE MSO: 250	0
3420,811	54- 81	NRRFYRQL [LLTA] DDRVNPC [IG] GV [ILFHE] PE: 72	2
4138,470	82-121	TLYQKADDGRFPQVIKSKGGVVGKVDKGVVPLAGTNGE	3
5244,080	82-132	[TL] YQKADDGRFPQVIKSKGGVVGKVDKG [VVPLAG] [TNG] E [TT TQG] [LDG] [LSE]	4
<i>β-Lactoglobulin (P02754)</i>			
684,393	62- 67	LKPTPE	0
800,488	106-112	NKVLVLD	0
929,545	72- 78	ILLQKWE	0
1098,568	62- 71	LKPTPEGDLE	2
1108,618	82- 90	CAQKKIIAE PE: 82	0
1218,676	17- 27	LIVTQTMKGLD	0
1234,671	17- 27	LIVTQTMKGLD MSO: 23	0
1361,706	50- 61	AQSAPLRVYVEE	0
1458,716	115-124	YKKYLLFCME PE: 122 MSO: 123	0
1657,819	131-143	QSLACQCLVRTPE PE: 135, 137	0
1674,791	113-124	TDYKKYLLFCME PE: 122 MSO: 123	1
1686,979	91-105	GTKIPAVFKIDALNE	1
1811,900	28- 44	IQKVAGTWYSLAMAASD	0
1903,017	45- 61	ISLLDAQSAPLRVYVEE	1
2069,971	115-130	YKKYLLFCMENSAEPE PE: 122	1
2115,984	131-137	QSLACQCLVRTPEVDDE	2
2323,295	154-173	KALKALPMHIRLSFNPTQLE MSO: 161	0
2456,260	106-124	NKVLVLDTDYKKYLLFCME PE: 122 MSO: 123	0
2697,490	151-173	KFDKALKALPMHIRLSFNPTQLE	1
2842,528	151-174	KFDKALKALPMHIRLSFNPTQLEE MSO: 161	1

3.2.1.3 Sequenzen aus Spaltungen mit Trypsin

Masse	Abschnitt	Sequenz	MC
<i>Aldolase (P00833)</i>			
641,352	56- 59	RFYR	1
646,367	312-317	AWGGKK	1
763,471	208-214	VLAADVYK	0
801,483	304-311	ALQASALK	0
845,434	134-139	CAQYKK PE:134	1
940,485	14- 21	ELSDIAHR	0
971,588	101-110	<u>GGVVG</u> IKVDK	1
975,460	201-207	CQYVTEK PE: 201	0
1065,511	140-148	DGADFAKWR	1
1068,580	13- 21	KELSDIAHR	1
1093,564	322-330	<u>AAQEE</u> YV <u>KR</u>	1
1130,632	312-321	AWGGKKENLK	2
1131,562	200-207	RCQYVTEK PE: 201	1
1180,614	331-341	ALANSLACQ <u>GK</u> PE: 338	0
1186,715	99-110	SKGGVVGIKVDK	2
1193,606	139-148	KDGADFAKWR	2
1332,700	28- 41	GILAADESTG <u>SI</u> AK	0
1342,711	87- 98	<u>ADDGR</u> PF <u>PQ</u> VIK	0
1428,832	304-317	ALQASALKAWGGKK	2
1469,786	1- 13	PHSHPALT <u>P</u> EQKK	1
1488,802	28- 42	<u>GILAA</u> DESTG <u>SI</u> AKR	1
1490,708	43- 55	<u>LQ</u> SIGTENT <u>EE</u> NR	0
1646,809	43- 56	<u>LQ</u> SIGTENT <u>EE</u> NR	1
1707,837	243-257	<u>Y</u> SHEEIAMATV <u>T</u> ALR MSO: 250	0
1802,910	42- 56	<u>RLQ</u> SIGTENT <u>EE</u> NR	2
1863,938	243-258	<u>Y</u> SHEEIAMATV <u>T</u> [AL] <u>RR</u> MSO:250	1
1898,059	22- 41	<u>I</u> VAPGKGILAADESTG <u>SI</u> AK	1
2054,160	22- 42	<u>I</u> VAPGKGILAADESTG <u>SI</u> AKR	2
2123,091	153-172	<u>I</u> GEHTPSALAIMENAN <u>V</u> LAR MSO:164	0
2242,041	342-363	YTPSGQAGAAASESLFISNHAY <u></u>	0
2270,463	111-133	<u>G</u> VVPLAGTNGETTTQGLDGL <u>SER</u>	1
2391,253	1- 21	PHSHPALT <u>P</u> EQKKELSDIAHR	2
2614,332	108-133	<u>V</u> DKGVVPLAGTNGETTTQGLDGL <u>SER</u>	1
3043,558	259-288	TVPPAVTGVTFLSGGQSEEEASINLNAINK	0
3068,540	173-199	YASICQQNGIVPIVEPEILPDGDHDLK PE:177	0
3224,721	101-133	GGVVGIKVDKGVVPLAGTNGETTTQGLDGLSER	2

Masse	Abschnitt	Sequenz	MC
<i>Schweineserum Albumin (P08835)</i>			
712,374	27- 32	SEIAHR	0
815,401	461-466	RPEEER	0
823,420	67- 73	EVTEFAK	0
829,525	450-457	KLGLVGSR	1
874,510	240-246	LSQRFPK	1
910,452	247-254	ADFTEISK	0
912,442	129-136	NDNPDIK	0
927,493	159-165	YLYEIAR	0
977,560	96-103	LCAIPSLR PE: 97	0
999,478	35- 42	DLGEQYFK	0
1017,536	87- 95	SIHTLFGDK	0
1072,567	398-406	FQPLVDEPK	0
1083,595	159-166	YLYEIARR	1
1115,519	411-418	QNCLEFEK PE: 413	0
1142,714	546-555	KQTALVELLK	1
1219,607	23- 32	DTYKSEIAHR	1
1274,642	33- 42	FKDLGEQYFK	1
1294,715	567-578	TVLGNFAAFVQK	1
1345,697	556-566	HKPHATEEQLR	0
1455,807	358-369	RHPDYSVSLLLR	1
1479,795	419-431	LGEYGFQNALIVR	0
1519,710	284-295	YICENQDTISTK PE: 286	0
1525,645	184-195	DVFSECCQAADK PE: 189, 190	0
1609,790	345-357	DVFLGTFLYEYSR	0
1623,943	436-450 oder 435-449	VPQVSTPTLVEVARK oder KVPQVSTPTLVEVAR	1
1650,895	247-261	ADFTEISKIVTDLAK	1
1700,868	467-480	LSCAEDYLSLVLNR PE: 469	0
2252,083	316-334	RDEL ^u PADLN [PLEHD] FVEDK	1
2320,216	166-183	RHPYFYAPELLYYAIYK	1
2614,253	385-406	EDPPACYATVFDK ^u FQPLVDEPK PE: 390	1

Masse	Abschnitt	Sequenz	MC
<i>Myoglobin (P02188)</i>			
735,488	79-102	HKIPK	1
748,435	134-139	ALFLFR	0
790,430	57- 63	ASEDLKK	1
941,473	146-153	YKELGFQG	1
1271,633	32- 42	LFTGHPETLEK	
1360,758	134-145	ALFLFRNDIAAK	1
1378,842	64- 77	HGTVVLTALGGILK	0
1506,937	64- 78	HGTVVLTALGGILKK	1
1518,664	119-133	HPGDFGADAQGAMTK MSO: 131	0
1553,796	140-153	NDIAAKYKELGFQG	2
1606,855	17- 31	VEADIAGHGQEVLR	0
1651,917	134-147	ALFLFRNDIAAKYK	2
1661,853	32- 45	LFTGHPETLEKFDK	1
1853,962	80- 96	GHHEAELKPLAQSHATK	1
1885,022	103-118	YLFISDAIHVLHSK	0
1937,017	32- 47	LFTGHPETLEKFDKFK	2
1982,057	79- 96	KGHHEAELKPLAQSHATK	1

3.2.1.4 Sequenzen aus Spaltungen mit Chymotrypsin

Masse	Abschnitt	Sequenz	MC
<i>Alkoholdehydrogenase (P00330)</i>			
574,335	88- 92	AGIKW	0
595,309	83- 87	KIGDY	0
636,372	35- 39	INVKY	0
640,313	190-195	AKAMGY	
729,306	93- 98	LNGSCM PE: 97	1
822,424	323-329	STLPEIY	0
825,425	306-312	DFFARGL	2
833,415	12- 18	YESHGKL	1
864,457	211-218	RSIGGEVF	0
881,455	186-193	AVQYAKAM	1
922,442	51- 58	HGDWPLPT 58: V->T FS : T58	1
930,537	259-267	VRANGTTVL	0
1021,641	21- 29	KDIPVPKPK FS : H20, K29	0
1039,687	313-322	VKSPIKVVL	0
1072,600	173-185	VAISGAAGGLGSL	1
1082,454	120-128	THDGSFQQY 127 : E->Q	1
1125,552	12- 20	YESHGKLEY 20 : H->Y	2
1172,688	222-232	TKEKDIVGAVL	0

Masse	Abschnitt	Sequenz	MC
<i>Alkoholdehydrogenase (P00330) – Fortsetzung</i>			
1201,628	295-305	VGNRADTREAL	0
1210,602	323-332	STLPEIYEKM	2
1239,637	211-221	RSIGGEVFIDF	1
1250,590	199-210	GIDGGEGKEELF	1
1260,683	1- 11	SIPETQKGVIF Acetyl: 1	0
1300,517	93-102	LNGSCMACEY PE: 97	1
1330,600	40- 50	SGVCHTDLHAW PE: 43	1
1340,660	298-308	RADTREALDFF FS: N297	2
1381,736	1- 12	SIPETQKGVIFY	1
1389,571	117-128	SGYTHDGSFQQY	2
1393,769	282-294	NQVVK SIS [IVG] SY	0
1423,737	330-341	EKMEKGQVGRY Heterogenität für V337: an Position 337 kommt zum Teil auch I vor (siehe unten)	1
1423,747	1- 12	SIPETQKGVIFY Acetyl: 1	1
1437,752	330-341	EKMEKGQIVGRY Heterogenität für I337: an Position 337 kommt zum Teil auch V vor (siehe oben)	1
1523,796	146-159	AQVAPILCAGITVY PE:153	1
1547,868	219-232	IDFTKEKDIVGAVL	1
1561,932	21- 34	KDIPVVPKPKANELL	1
1594,721	268-281	VGMPAGAKCCSDVF PE: 276, 277	0
1610,792	295-308	VGNRADTREALDFF	2
1618,843	196-210	RLGIDGGEGKEELF	2
1711,720	103-116	CELGNESNC [PH] ADL PE: 103, 111	
1854,038	19- 34	EYKDIPVPK [PK] [AN] ELL 20: H->Y	2
1938,949	63- 82	VGGHEGAGVVVGMGENVKGW	1
2018,837	103-119	CELGNESNC [PH] ADLSGY PE: 103, 111	2
2204,186	59- 81	KLPLVGGHEG [AG] V [VVGMGE] NVKG FS: V58, G81	3
2241,158	323-341	STLPEIYEKMGQIVGRY	2
2282,930	99-116	ACEYCELGNESNCPHADL PE: 100, 103, 111	2
2358,231	129-152	ATADAVQAAHI PQGTDLAEVAPVL 151: I->V 147: Q->E	1
2393,307	211-232	RSIGGEVFIDFTKEKDIVGAVL	2

Masse	Abschnitt	Sequenz	MC
<i>Alkoholdehydrogenase (P00330) – Fortsetzung</i>			
2590,048	99-119	ACEYCELGNESNCPHADLSGY PE: 100, 103, 111	3
2603,314	129-155	ATADAVQAAHIPQGTDLAEVAPVLC [AA] C liegt reduziert vor ! 155: G->A 151: I->V 147: Q->E	2
3184,593	129-159	ATADAVQAAHIPQGTDLAEVAPVLC AATVY C liegt reduziert vor ! 155: G->A 151: I->V 147: Q->E	2
<i>Aldolase (P00833)</i>			
586,403	328-332	VKRAK	0
688,424	204-209	VTEKVL	0
704,336	358-363	ISNHAY	0
774,360	337-342	ACQGKY PE: 338	0
780,389	138-144	KKDGADF	0
802,403	199-203	KRCQY PE: 201	0
838,394	321-327	KAAQEEY	0
961,420	131-137	SECAQY PE :134	0
974,498	162-170	AIMENAMVL	1
977,524	284-291	NAINKCPL PE : 289	0
980,550	145-151	AKWRCVL PE :149	1
1023,668	21- 30	[RIV] APGKGIL FS : H20	
1049,537	31- 41	AADESTGSI AK FS: K41	1
1052,573	152-161	KIGEHTPSAL	0
1083,558	214-222	KALSDHHIY	1
1149,538	116-127	AGTNGETTTQGL	1
1159,556	333-342	ANSLACQGKY PE: 338	1
1165,600	138-147	KKDGADFAKW	1
1196,642	214-223	KALSDHHIYL	2
1246,552	128-137	DGLSERCAQY PE: 134	1
1309,636	233-243	VTPGHACTQKY PE: 239	0
1318,696	31- 43	AADESTGSI AKRL	0
1320,591	271-283	SGGQSEEEASINL	0
1388,673	244-256	SHEEIAMATVTAL MSO: 250	1

Masse	Abschnitt	Sequenz	MC
<i>Aldolase (P00833) – Fortsetzung</i>			
1393,659	343-357	<u>TPSGQAGAA [AS] ESLF</u>	1
1400,801	257-269	<u>RRTVPPAVTGVTF</u>	0
1416,658	126-137	<u>GLDGLSERCAQY</u> PE: 134 FS: Q125	2
1421,681	84- 94	<u>YQKADDGRPFQ</u> FS: Q94	2
1488,709	229-241	<u>KPNMVTPGHACTQ</u> PE: 239 FS: Q228	1
1564,797	314-327	<u>GGKKENLKAAQEEY</u>	1
1647,852	64- 78	<u>TADDRVNPC [IG] GVIL</u> PE: 72	0
1652,011	99-115	<u>SKGGVVGIKVDKGVVPL</u> FS: K98	0
1663,768	44- 57	<u>pESIGTENTEENRRF</u> 44: Q->pyro-E (nach Endopeptidasespaltung !)	0
1680,794	44- 57	<u>QSIGTENTEENRRF</u>	0
1709,922	214-228	<u>K [AL] SDHHIYLEGTLL</u>	4
1711,913	1- 15	<u>PHSHPALTPEQKKEL</u>	1
1716,805	45- 58	<u>SIGTENTEENRRFY</u> FS: Q44	1
1760,936	63- 78	<u>[LT] ADDRVNPC [IGG] VIL</u> PE: 72	1
1779,867	229-243	<u>KPNMVTPGHAC [TQ] KY</u> PE: 239	1
1794,921	64- 79	<u>TADDRVNPCIGGVILF</u> PE: 72	1
1827,905	44- 58	<u>pESIGTENTEENRRFY</u> 44: E->pyro-E (nach Endopeptidasespaltung !)	1
1909,005	63- 79	<u>LTADDRVNPCIGG [VI] LF</u> PE: 72	2
2021,089	62- 79	<u>LLTADDRVNPCIGGVILF</u> PE: 72	3
2054,161	21- 41	<u>RIVAPGKGILAA [DE] STGSIK</u> FS: K41	1
2235,152	1- 20	<u>PHSHPALTPEQKKELSDIAH</u> FS: H20	2
2377,073	116-137	<u>AGTNGETTTQGLDGL [SE] RCAQY</u> PE: 134	2
2711,387	181-203	<u>GIVPIVEPEILPDGDHDLKRCQY</u> PE: 201 FS: N180	2
3560,766	174-203	<u>A [SI] CQNGIVPIVEPEILPDGDHDLKRCQY</u> PE: 177, 201	1

3.2.1.5 Sequenzen aus Spaltungen mit Endoproteinase AspN

Masse	Position	Sequenz	MC
<i>DrTI (Serin Proteinase Inhibitor)</i>			
811,09	1- 7	SDAEKVY	1
1231,86	154-164	EGRRSLVLKSS FS: E153	0
1247,40	99-111	DSGEARVAIGGSE	0
1448,57	76- 87	DTELEIEFVEKP	1
2152,87	166-184	DSPFRVVFVKPRSGSETES	0
2240,27	165-184	DDSPFRVVFVKPR [SG] SETES	1
2499,25	54- 75	DLQMGLPVRFSSPEESQGKIYT	0
4923,427	8- 53	DIEGYPVFLGSEYYIVSAIIIGAGGGVVRPGRTRGSMCPMS [II] QEQS PE: 44	0

3.2.2 Grafische Darstellung der Sequenzierungsergebnisse – Sequenzalignments

Die nachfolgenden Sequenzalignments veranschaulichen die Sequenzabdeckungen durch die im vorangegangenen Punkt aufgelisteten Peptide für die im einzelnen sequenzierten Proteine. Die Zeile „Sequenz“ gibt die entsprechende Sequenz des Proteins gemäß SWISS-PROT Datenbank wider (siehe oben; Ausnahme: *DrTI*). Die unter den Sequenzen liegenden Farbcodierungen gelten jeweils für die in der linken Spalte angegebene Endoproteinasespaltung des Proteins und sind wie folgt zu interpretieren:

❖ Schwarz unterlegte Bereiche:

Diese Sequenzbereiche wurden durch wiedergefundene Spaltfragmente des Proteins nach deren RP-HPLC Trennung noch abgedeckt.

❖ Rot / blau unterlegte Bereiche:

Diese Sequenzabschnitte konnten bei C-terminaler Sequenzierung (rot) bzw. N-terminaler Sequenzierung (blau) eines Proteinfragments der angegebenen Endoproteinasespaltung lückenlos sequenziert werden.

❖ Rosa / hellblau unterlegte Bereiche:

Diese Sequenzabschnitte konnten bei C-terminaler Sequenzierung (rosa) bzw. N-terminaler Sequenzierung (hellblau) eines Proteinfragments der angegebenen Endoproteinasespaltung nur in der Summe abgebaut werden, entsprechen also einer Sequenzlücke beim Abbau (Gap).

❖ Grün unterlegte Bereiche:

Diese Sequenzabschnitte entsprechen einer Überlappung einer C-terminal erhaltenen Sequenz eines Proteinfragments und mit einer N-terminal erhaltenen Sequenz innerhalb des gleichen Proteinfragments der angegebenen Endoproteinaspaltung.

❖ Grau markierte Bereiche der Sequenz:

Die Summe der C- und N-terminal eindeutig erhaltenen Sequenzbereiche (keine Sequenzlücken) ist hellgrau dargestellt. Die Summe umfasst sequenzierte Fragmente aus allen Endoproteinaspaltungen eines Proteins, sofern mehrere Endoproteinaspaltungen durchgeführt wurden. Dunkelgrau markierte, kursiv gedruckte Bereiche der Sequenz weisen auf Sequenzlücken in der Summe aller durchgeführten Sequenzierungen des Proteins hin.

❖ Bedeutung weiterer, speziell hervorgehobener Sequenzpositionen:

- a: **a**cetylierter N-Terminus
- x: Aminosäureaustausch (exchange)
- p: **p**hosphorylierte Aminosäure
- g: **g**lycosilierte Aminosäure
- r: **r**eduziertes Cystein (liegt sonst als pyridinethyliertes Derivat vor)
- h: **h**eterogene Stelle in der Sequenz

Alkoholdehydrogenase (P00330) – nur C-terminale Sequenzierung

AS-Nummer	10	20	30	40	50	60
Sequenz	SIPETQKG	VIFYESHG	KLEHKDIP	VPKPKANELL	INVKYS	GVCHTDLHAWHGDWPLPVKL
Chymotrypsin						
AS-Nummer	70	80	90	100	110	120
Sequenz	PLVGGHEGAG	VVVMGENV	KGWKIGDY	AGIKWLNG	SCMACEY	CELGNESNC
Chymotrypsin						
AS-Nummer	130	140	150	160	170	180
Sequenz	HDGSFQQY	ATADAVQA	AHIPQGTDL	AQVAPILCAA	ITVYKALK	SANLMAGHWVAISGAAG
Chymotrypsin						
AS-Nummer	190	200	210	220	230	240
Sequenz	GLGSLAVQY	AKAMGYRVL	GIDGGEGKE	ELFRSIGGE	VFIDFTKEK	DIVGAVLKATDGGAH
Chymotrypsin						
AS-Nummer	250	260	270	280	290	300
Sequenz	GVINVS	VSEAAIEA	STRYVRANG	TTVLVGM	PAGAKCCS	DFNQVVK
Chymotrypsin						
AS-Nummer	310	320	330	340	347	
Sequenz	TREALDFF	ARGLVKSP	IKVVGLSTL	PEIYEKME	KGQIVGRY	VVDTSK
Chymotrypsin						

Cytochrom C (P00004) – nur C-terminale Sequenzierung

AS-Nummer	10	20	30	40	50	60
Sequenz	a GDVEKGKKIFVQKCAQCHTVEKGGKHKTGPNLHGLFGRKTGQAPGFTYTDANKNKGITWK					
LysC						
AS-Nummer	70	80	90	100	104	
Sequenz	EETLMEYLENPKKYIPGTKMIFAGIKKTEREDLIAYLKATNE					
LysC						

 α -Casein (P02662)

AS-Nummer	10	20	30	40	50	60
Sequenz	MKLLILTCLVAVALARPKHPIKHQGLPQEVLENENLLRFFVAPFPEVFGKEKVNELSKDIG					
GluC	Signal (1-15)					
AS-Nummer	70	80	90	100	110	120
Sequenz	p p SESTEDQAMEDIKQMEAESISSSEEIVPNSVEQKHQKEDVPSELYLGYLEQLLRLLKYYK					
GluC						
AS-Nummer	130	140	150	160	170	180
Sequenz	p VPQLEIVPNSAEERLHSMKEGIHAQQKEPMIGVNQELAYFYPELFRQFYQLDAYPSGAWY					
GluC						
AS-Nummer	190	200	210	214		
Sequenz	x YVPLGTQYTDAPSFSDIPNPIGSENSEKTMPLW					
GluC						

 β -Casein (P02666)

AS-Nummer	10	20	30	40	50	60
Sequenz	MKVLILACLVALALARELEELNVPGEIVESLSSEESITRINKKIEKFQSEEQQQTEDEL					
LysC	Signal (1-15)					
AS-Nummer	70	80	90	100	110	120
Sequenz	g g QDKIHFPFAQTQSLVYPFPGPIPNLQNIPLTQTPVVVPPFLQPEVMGVSKVKEAMAPK					
LysC						
AS-Nummer	130	140	150	160	170	180
Sequenz	HKEMFPFKYPVEPFTESSQLTLTDVENLHLPLPLLSWMHQPHQPLPPTVMFPQSVLSL					
LysC						
AS-Nummer	190	200	210	220	224	
Sequenz	g g SQSKVLPVPQKAVPYPQRDMP IQAFLLYQEPVLGPVRGPFPIIV					
LysC						

Rinderserum Albumin (P02769)

AS-Nummer	10	20	30	40	50	60
Sequenz	MKWVTFISLLLLFSSAYSRGVFRRDTHKSEIAHRFKDLGEEHFKGLVLIAFSQYLQQCPF					
LysC	Signal (1-18)					
AS-Nummer	70	80	90	100	110	120
Sequenz	DEHVKLVNELTEFAKTCVADESHAGCEKSLHTLFGDELCKVASLRETYGDMADCCEKQEP					
LysC						
AS-Nummer	130	140	150	160	170	180
Sequenz	ERNECFLSHKDDSPDLPKLKPDPNTLCDEFKADEKKFWGKLYEIAARRHPYFYAPELLYY					
LysC						
AS-Nummer	210	200	210	220	230	240
Sequenz	ANKYNGVFQEECCQAEDKGACLLPKIETMREKVLASSARQRLRCASIQKFGERALKAWSVA					
LysC						
AS-Nummer	250	260	270	280	290	300
Sequenz	RLSQKFPKAEFVEVTKLVTDLTQVHKECCHGDLLECADDRADLAKYICDNQDTISSKLKE					
LysC						
AS-Nummer	310	320	330	340	350	360
Sequenz	CCDKPLLEKSHCIAEVEKDAIPENLPPLTADFAEDKDVCKNYQEAKDAFLGSFLYEYSRR					
LysC						
AS-Nummer	370	380	390	400	410	420
Sequenz	HPEYAVSVLLRLAKEEYEATLEECACDDPHACYSTVFDKCLKHLVDEPQNLIKQNCDFEK					
LysC						
AS-Nummer	430	440	450	460	470	480
Sequenz	LGEYGFQNALIVRYTRKVPQVSTPTLVEVSRLGKVGTRCCTKPESERMPCTEDYLSLIL					
LysC						
AS-Nummer	510	500	510	520	530	540
Sequenz	NRLCVLHEKTPVSEKVTKCCTESLVNRRPCFSALTPDETYVPKAFDEKLFTFHADICTLP					
LysC						
AS-Nummer	570	560	570	580	590	600
Sequenz	DTEKQIKKQTALVELLKHKPKATEEQLKTMENFVAFVDKCCAADDKEACFAVEGPKLVV					
LysC						
AS-Nummer	607					
Sequenz	STQTALA					
LysC						

Schweineserum Albumin (P08833)

AS-Nummer	10	20	30	40	50	60
Sequenz	WVTFISLLFLFSSAYS	SRGVFR	RDYKSEIAHR	FKDLGEQYFKGLVLIAFS	QHLQQCPYEE	
Trypsin						
AS-Nummer	70	80	90	100	110	120
Sequenz	HVKLVREVTEFAKTC	VADESAENCDKSI	HTLFGDKLCAIP	SLREHYGDLAD	CCEKEEPER	
Trypsin						
AS-Nummer	130	140	150	160	170	180
Sequenz	NECFLOHKNDNPD	IPKLKPDFVALC	ADFQED	EQKFWGKYL	EIARRHPYFYA	PELLLYAI
Trypsin						
AS-Nummer	190	200	210	220	230	240
Sequenz	IYKDVFSECCQA	ADKAACLLPK	IEHLREKVL	TSAKQRLK	CASIQKFG	ERAFKAWSLARL
Trypsin						
AS-Nummer	250	260	270	280	290	300
Sequenz	SQRFPAKADFTE	ISKIVTDLAK	VHKECCHG	DLLECADDR	ADLAKYIC	ENQDTISTKLKECC
Trypsin						
AS-Nummer	310	320	330	340	350	360
Sequenz	DKPLLEKSHCIA	EAKRDELPA	DLNPLEHDF	VEDKEVCK	NYKEAKDV	FLGTFLYEYSRRHP
Trypsin						
AS-Nummer	370	380	390	400	410	420
Sequenz	DYSVSLLLRI	AKIYEATLE	DCCAKEDPP	PACYATV	FDKFQPLV	DEPKNLIKQNC
Trypsin						
AS-Nummer	430	440	450	460	470	480
Sequenz	EYGFQNALIV	RYTKKVPQ	VSTPTLVE	VARKLGLV	GSRCCKR	PEEEERLS
Trypsin						
AS-Nummer	490	500	510	520	530	540
Sequenz	LCVLHEKTPV	SEKVTKCCT	ESLVNRRP	CFSALTP	DETYKPKE	FVEGTFTFHADLCTLPED
Trypsin						
AS-Nummer	550	560	570	580	590	600
Sequenz	EKQIKKQTAL	VELLKHKP	HATEEQ	LRTVLGN	FAAFVQK	CCAAPDHEACFAVEGPKFVIEI
Trypsin						
AS-Nummer	605					
Sequenz	RGILA					
Trypsin						

Myoglobin

AS-Nummer	10	20	30	40	50	60
Sequenz	GLSDGEWQQVLNVWGKVEADIAHGQEVLRIRLFTGHPETLEKFDKFKHLKTEAMKASED					
LysC						
Trypsin						
AS-Nummer	70	80	90	100	110	120
Sequenz	LKKHGTVVLTALGGILKKKGHHEAELKPLAQSHATKHKIPIKYLEFISDAIIHVLHSHKHP					
LysC						
Trypsin						
AS-Nummer	130	140	150			
Sequenz	GDFGADAQQGAMTKALELFRNDIAAKYKELGFQG					
LysC						
Trypsin						

DrTI (Serin Proteinase Inhibitor)

AS-Nummer	10	20	30	40	50	60
Sequenz	SDAEKVYDIEGYPVFLGSEYYIVSAIIGAGGGVVRPGRTRGSMCPMSIIQEQLQMGLP					
LysC						
AspN						
AS-Nummer	70	80	90	100	110	120
Sequenz	VRFSSPEESQGKIYTDTELEIEFVEKPDCAESSKWIKDSGEARVAIGGSSEDPHQGELVR					
LysC						
AspN						
AS-Nummer	130	140	150	160	170	180
Sequenz	GFFKIEKLGSLAYKLVFCPKSSSGSCSDIGINYEGRRSLVLKSSDDSPFRVVFVKPRSGS					
LysC						
AspN						
AS-Nummer	184					
Sequenz	ETES					
LysC						
AspN						

β-Lactoglobulin

AS-Nummer	10	20	30	40	50	60
Sequenz	MKCLLLALALTCGAQALIVTQTMKGLDIQKVAGTWYSLAMAASDISLLDAQSAPLRVYVE					
LysC						
GluC						
AS-Nummer	70	80	90	100	110	120
Sequenz	ELKPTPEGDLEILLQKWENGCAQKKIIAEKTKIPAVFKIDALNENKVLVLDTDYKKYLL					
LysC						
GluC						
AS-Nummer	130	140	150	160	170	178
Sequenz	FCMENSAEPEQSLACQCLVRTPEVDDEALEKFDKALKALPMHIRLSFNPTQLEEQCHI					
LysC						
GluC						

Aldolase

AS-Nummer	10	20	30	40	50	60
Sequenz	PHSHPALTPEQKKELSDIAHRIVAPGKGILAADESTGSIAKRLQSIGTENTENRRFYRQ					
LysC						
GluC						
Trypsin						
Chymotrypsin						
AS-Nummer	70	80	90	100	110	120
Sequenz	LLLTADDRVNPCIGGVILFHETLYQKADDGRFPFQVIKSKGGVVGIVDKGVVPLAGTNG					
LysC						
GluC						
Trypsin						
Chymotrypsin						
AS-Nummer	130	140	150	160	170	180
Sequenz	ETTTQGLDGLSERCAQYKKDGADFAKWRCVLKIGEHTPSALAIMENANVLARYASICQQN					
LysC						
GluC						
Trypsin						
Chymotrypsin						
AS-Nummer	190	200	210	220	230	240
Sequenz	GIVPIVEPEILPDGDHDLKRCQYVTEKVLAAVYKALSDHHIYLEGTLKPNMVTTPGHACT					
LysC						
GluC						
Trypsin						
Chymotrypsin						
AS-Nummer	250	260	270	280	290	300
Sequenz	QKYSHEEIAMATVTALRRTPPAVTGVTFLSGGQSEEEASINLNAINKCPLLKPWALTFS					
LysC						
GluC						
Trypsin						
Chymotrypsin						
AS-Nummer	310	320	330	340	350	360
Sequenz	YGRALQASALKAWGGKKENLKAAQEEYVKRALANSLACQGYTPSGQAGAAASESLFISN					
LysC						
GluC						
Trypsin						
Chymotrypsin						
AS-Nummer	363					
Sequenz	HAY					
LysC						
GluC						
Trypsin						
Chymotrypsin						

3.2.3 Statistische Auswertung der Sequenzierungsergebnisse

3.2.3.1 Definition des sequenzierbaren Anteils

Aus theoretischer Sicht ergibt sich der sequenzierbare Anteil eines Proteins für die Leitersequenzierung durch die Summe der Aminosäuren aller Spaltpeptide einer Endopeptidasespaltung mit einer Masse über 700 Da, bezogen auf die Gesamtsequenz. Für eine vereinfachende theoretische Betrachtung wird davon ausgegangen, dass die Endopeptidase keine mögliche Spaltungsstelle im Protein auslöst, d.h. MC = 0. Tabelle 31 zeigt, dass der theoretisch sequenzierbare Anteil dann im Mittel über alle untersuchten Spaltungen und Proteine gesehen bei ca. 84% liegt.

Tabelle 31: Sequenzierbare Anteile der untersuchten Proteine bei Spaltung mit unterschiedlichen Endopeptidasen

Protein	AS im Protein ¹	Fragmente über 700 Da (theoretisch)	HPLC-Fragmente (praktisch)	HPLC-Fragmente über 700 Da (praktisch)
<i>Endoproteinase LysC</i>				
Cytochrom C	104	63%	84%	79%
Myoglobin	153	80%	72%	72%
β-Lactoglobulin	162	83%	91%	88%
<i>DrTI (Serin Proteinase Inhibitor)</i>	184	93%	95%	95%
β-Casein	209	94%	83%	83%
Aldolase	363	93%	94%	94%
Rinderserum Albumin	589	92%	64%	64%
<i>Mittelwert</i>		85%	83%	82%
<i>Endoproteinase GluC (im Phosphatpuffer)</i>				
β-Lactoglobulin	162	73%	91%	91%
α-Casein	199	74%	90%	88%
Aldolase	363	92%	100%	100%
<i>Mittelwert</i>		80%	94%	93%
<i>Trypsin</i>				
Myoglobin	153	76%	84%	84%
Aldolase	363	87%	80%	79%
Schweineserum Albumin	589	82%	49%	49%
<i>Mittelwert</i>		81%	71%	70%
<i>Chymotrypsin</i>				
Alkoholdehydrogenase	347	74%	87%	84%
Aldolase	363	74%	86%	85%
<i>Mittelwert</i>		74%	87%	84%
<i>Endoproteinase AspN</i>				
<i>DrTI (Serin Proteinase Inhibitor)</i>	184	99%	71%	71%
<i>alle Spaltungen</i>				
<i>Mittelwert</i>		84%	79%	78%

¹ Die Aminosäuren der Signalsequenz wurden hier bereits abgezogen !

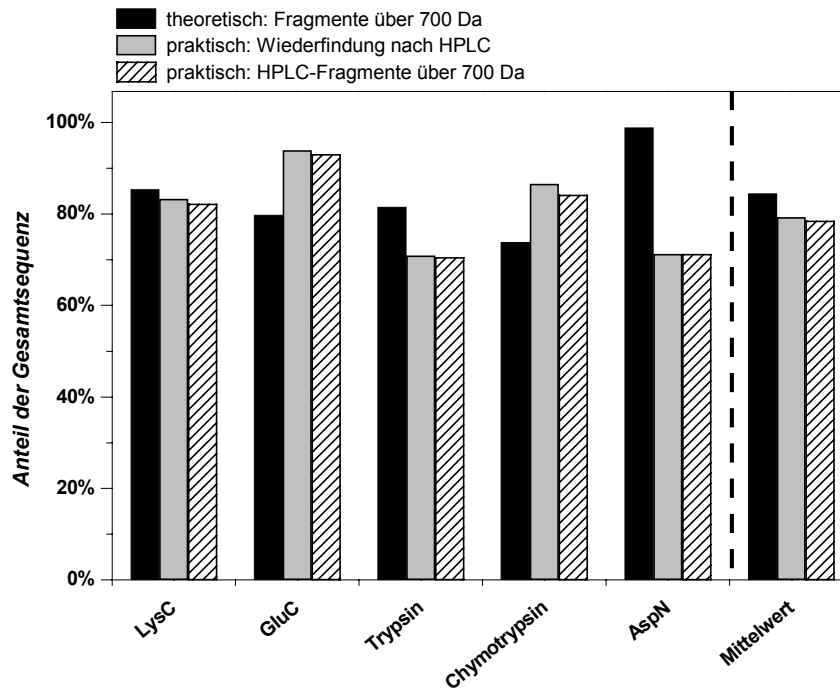
In der Praxis hängt der sequenzierbare Anteil jedoch auch noch davon ab, welcher Anteil der Spaltpeptide nach der HPLC-Trennung wiedergefunden wird. Tabelle 31 zeigt, dass durch die RP-HPLC Trennung die Sequenzabdeckung durch die wiedergefundenen Peptidfragmente im Mittel nochmals um etwa 5% auf 79% abnimmt. Dabei nimmt auf den ersten Blick jedoch überraschend die Sequenzabdeckung nicht für alle Proteine ab. Im Falle der Spaltungen mit Endoproteinase GluC, wie auch bei Spaltungen mit Chymotrypsin ist der sequenzierbare Anteil durch wiedergefundene Fragmente nach der RP-HPLC sogar größer, als der theoretisch sequenzierbare Anteil durch Fragmente mit einer Masse über 700 Da. Diese Tatsache lässt sich durch die im Realfall auftretenden Fragmente mit ausgelassenen Spaltungsstellen erklären ($MC \neq 0$). Dadurch können Fragmente, die eigentlich eine Masse kleiner als 700 Da besitzen, in Form größerer Fragmente mit einer Masse über 700 Da auftreten. Gerade bei Spaltungen mit Endoproteinase GluC (Phosphatpuffer) und Chymotrypsin, aber auch mit Trypsin, treten solche *missed cleavages* häufig auf. Wie ein Blick auf die Sequenzen unter 3.2.1.2 und 3.2.1.4 verdeutlicht, werden bei vielen Peptiden ein oder zwei ausgelassene Spaltungsstellen beobachtet. Diese ausgelassenen Spaltungsstellen wirken sich somit also bei Proteinspaltungen diesen beiden Proteasen positiv auf die Sequenzabdeckung aus.

Der aus praktischer Sicht für die Sequenzierung zugängliche Anteil eines Protein über diese HPLC Fragmente verringert sich dann noch um den Anteil von Fragmenten mit einer Masse unter 700 Da. Diese kleinen Fragmente werden jedoch zumeist auch nach der RP-HPLC nicht wiedergefunden. Sie sind oft entweder zu hydrophil und eluieren nahezu bei der Totzeit oder sie zeigen bei der Detektion eine zu geringe UV-Absorption. Der sequenzierbare Proteinanteil durch wiedergefundene HPLC-Fragmente mit einer Masse über 700 Da ist daher wie Tabelle 31 zeigt mit 78% auch nur unbedeutend geringer als der insgesamt nach HPLC wiedergefundene Proteinanteil (79%). Abbildung 30 (folgende Seite) gibt die Resultate graphisch wieder.

Man sieht, dass sowohl der theoretisch als auch der praktische sequenzierbare Anteil der Gesamtsequenz für alle Endopeptidase-Spaltungstypen sehr ähnlich ist. Eine Diskrepanz ergibt sich jedoch für die Spaltungen mit Endoproteinase AspN im Vergleich zu den Spaltungen mit Endoproteinase GluC (Phosphatpuffer) oder Chymotrypsin. Aus theoretischer Sicht wäre eine Spaltung mit Endoproteinase AspN den beiden anderen Spaltungstypen vorzuziehen. Jedoch ist der praktisch sequenzierbare Anteil, der sich durch die Wiederfindung nach der HPLC ergibt für die Spaltung mit Endoproteinase GluC (Phosphatpuffer) und Chymotrypsin höher als bei der Spaltung mit Endoproteinase AspN. Dieses Phänomen basiert zum einen auf den bereits oben erwähnten *missed cleavages*, zum anderen aber auch auf der Tatsache, dass bei der Spaltung mit Endoproteinase AspN sehr viele der großen und damit oft

sehr hydrophoben Spaltfragmente bei der HPLC-Trennung offensichtlich nicht wieder von der Säule eluieren. Durch ein oder zwei solcher großen Fragmente, die nach der HPLC nicht wiedergefunden werden, verringert sich der sequenzierbare Anteil natürlich drastisch.

Abbildung 30: Sequenzierbare Anteile für verschiedene Endopeptidasespaltungen



Demzufolge erweisen sich mit Spaltungen mit den Endoproteinasen GluC (im Phosphatpuffer) und Chymotrypsin auch solche Endopeptidasespaltungen, die theoretisch zunächst als weniger günstig bezüglich des sequenzierbaren Anteils erscheinen, in der Praxis als vorteilhaft. Spaltungen mit Endoproteinasen LysC haben in der Theorie, wie auch in der Praxis einen relativ hohen sequenzierbaren Anteil. Dagegen bleiben Endoproteinasespaltungen mit Trypsin und AspN in der Praxis im Bezug auf den sequenzierbaren Anteil hinter den theoretischen Erwartungen zurück.

3.2.3.2 Sequenzabdeckungen der untersuchten Proteine

Die folgende Tabelle zeigt die ermittelten Werte für Sequenzabdeckungen der untersuchten Proteine bei unterschiedlichen Endopeptidasespaltungen. Diese Sequenzabdeckungen wurden aus den experimentell erhaltenen Teilsequenzen unter Punkt 3.2.1 errechnet. Dargestellt sind die Werte für die Sequenzabdeckung bezogen auf die Gesamtsequenz (GS) und auf den praktisch sequenzierbaren Anteil der Sequenz (SA).

Tabelle 32: Werte für die Sequenzabdeckungen verschiedener Proteine durch enzymatische Leitersequenzierung nach unterschiedlichen Endopeptidasespaltungen

Protein	N-terminal		C-terminal		Gesamt	
	GS ¹	SA ²	GS ¹	SA ²	GS ¹	SA ²
<i>Endoproteinase LysC</i>						
Cytochrom C	- ³	- ³	30%	38%	30%	38%
Myoglobin	9%	13%	15%	21%	24%	34%
β-Lactoglobulin	11%	12%	12%	14%	23%	26%
<i>DrTI</i>	19%	20%	26%	27%	45%	47%
β-Casein	6%	8%	15%	19%	22%	27%
Aldolase	6%	6%	20%	21%	26%	28%
Rinderserum Albumin	16%	25%	11%	18%	28%	43%
<i>Mittelwert</i>	11%	14%	18%	22%	28%	32%
<i>Endoproteinase GluC (im Phosphatpuffer)</i>						
β-Lactoglobulin	30%	33%	24%	26%	56%	61%
α-Casein	21%	23%	19%	22%	41%	46%
Aldolase	19%	19%	39%	39%	50%	50%
<i>Mittelwert</i>	23%	25%	27%	29%	49%	52%
<i>Trypsin</i>						
Myoglobin	14%	16%	21%	25%	35%	41%
Aldolase	10%	13%	11%	14%	21%	27%
Schweineserum Albumin	4%	7%	3%	6%	7%	13%
<i>Mittelwert</i>	9%	12%	12%	15%	21%	27%
<i>Chymotrypsin</i>						
Alkoholdehydrogenase	- ³	- ³	28%	34%	28%	34%
Aldolase	16%	19%	25%	29%	41%	48%
<i>Mittelwert</i>	- ⁴	- ⁴	26%	31%	35%	41%
<i>Endoproteinase AspN</i>						
<i>DrTI</i>	3%	4%	13%	18%	16%	22%
<i>Mittelwert</i>	13%	16%	20%	23%	31%	37%

¹ GS: Sequenzabdeckung bezogen auf die gesamte Proteinsequenz (abzüglich einer eventuellen Signalsequenz).

² SA: Sequenzabdeckung bezogen auf den praktisch sequenzierbaren Anteil – also wiedergefundene Proteinfragmente nach RP-HPLC mit einer Masse über 700 Da.

³ für diese Proteinfragmente liegen keine N-terminalen Sequenzen vor

⁴ nur ein Wert – keine Mittelwertbildung möglich

Bei den untersuchten Proteinen besteht, wie bei den synthetischen Peptiden, eine geringfügige Differenz zwischen der Zahl der abgebauten Aminosäuren und der Zahl der tatsächlich sequenzliefernden Aminosäuren. Die Werte in Tabelle 32 beziehen sich ausschließlich auf die Zahl derjenigen Aminosäuren, die auch tatsächlich Information über die Sequenzabfolge liefern. Der durch die Exopeptidasen über alle Sequenzierungsversuche im Mittel abgebaute

Anteil der Proteinsequenzen liegt hier, wie auch bei den synthetischen Peptiden um ca. 3% höher als derjenige Anteil, der tatsächlich Sequenzinformation liefert. Dieses Resultat bestätigt die Diskrepanz die zwischen Abbaulänge und Sequenzlänge die bereits bei den synthetischen Peptiden festgestellt wurde (ebenfalls ca. 3% zwischen Abbaulänge und Sequenzlänge).

Die nachfolgenden drei Abbildungen veranschaulichen die Ergebnisse aus Tabelle 32 grafisch. Die Spaltungen mit den Endoproteinasen GluC und Chymotrypsin, die bereits den höchsten sequenzierbaren Anteil gezeigt haben, liefern auch die besten Resultate für die praktisch erzielte Sequenzabdeckung. Wie Abbildung 31 deutlich zeigt, liegen bei Spaltungen mit den Endoproteinasen GluC und Chymotrypsin die Sequenzabdeckungen im Bezug auf den theoretisch sequenzierbaren Anteil höher als im Bezug auf den praktisch sequenzierbaren Anteil. Dieser scheinbare Widerspruch resultiert aus der Vielzahl von Peptiden mit ausgelassenen Spaltungsstellen (*missed cleavages*) bei diesen Endopeptidasespaltungen. Damit verbundenen ist die Tatsache, das der praktisch sequenzierbare Anteil nach RP-HPLC durch Fragmente mit Massen größer 700 Da höher ist, als der theoretisch sequenzierbare Anteil (siehe Tabelle 31 und Abbildung 30).

Abbildung 31: Gesamtsequenzabdeckungen bei verschiedenen Endopeptidasespaltungstypen

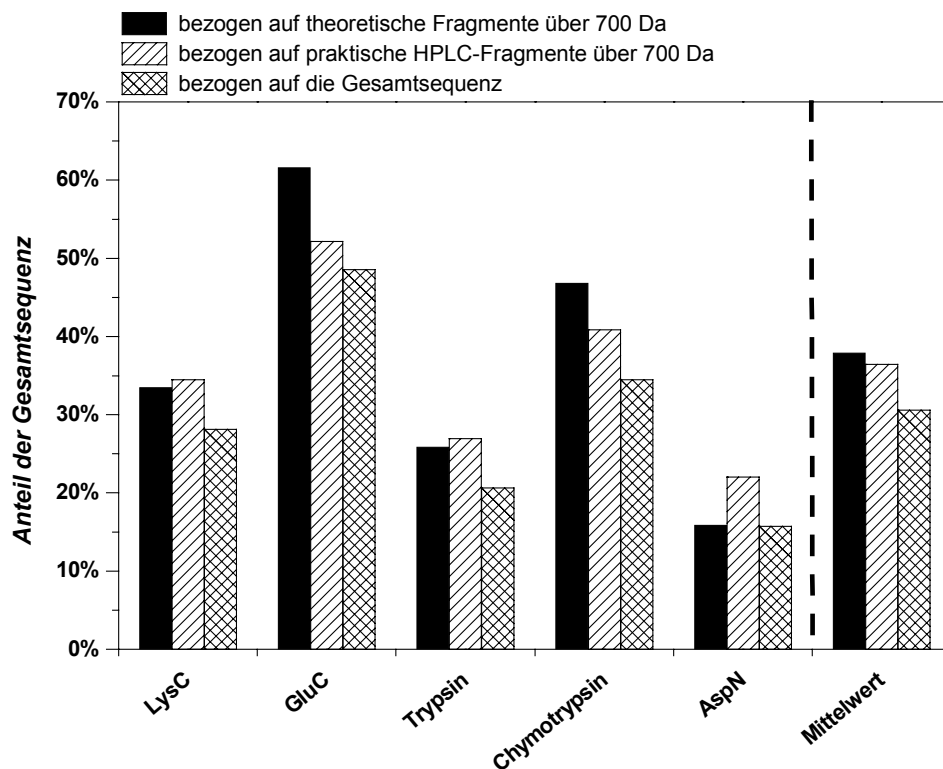


Abbildung 32: N-terminale Sequenzabdeckungen bei verschiedenen Endopeptidasespaltungstypen

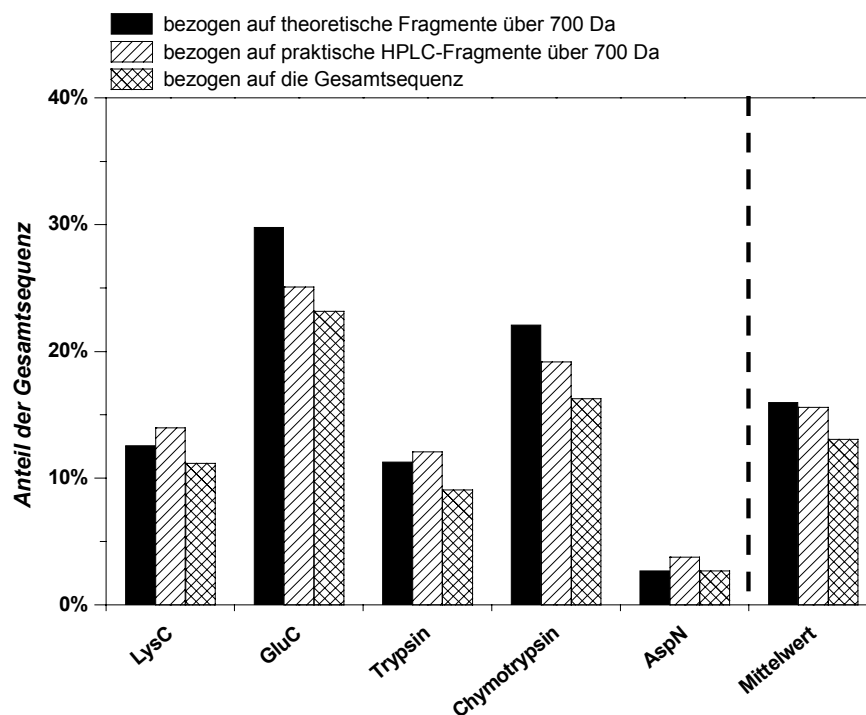
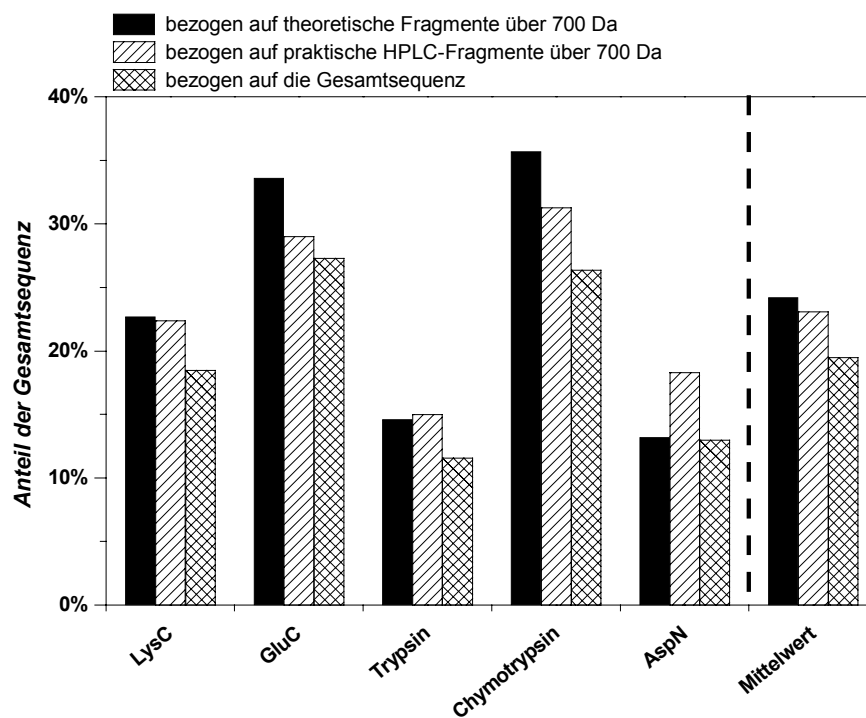


Abbildung 33: C-terminale Sequenzabdeckungen bei verschiedenen Endopeptidasespaltungstypen



3.2.3.3 Sequenzlängenverteilung in Abhängigkeit der Endopeptidasespaltung

Im folgenden wurde ausgewertet, mit welcher Häufigkeit bestimmte Sequenzlängen bei den Sequenzierungen der Spaltfragmente aus unterschiedlichen Endopeptidasespaltungen auftreten. Tabelle 33 zeigt zunächst, das von den insgesamt unter Punkt 3.2.1 aufgelisteten Spaltfragmenten im Mittel gerade einmal um 50% überhaupt eine Sequenzinformation lieferten, d.h. es konnte mindestens eine Aminosäure enzymatisch abgebaut werden.

Tabelle 33: Anteile der gefundenen Spaltfragmente, die für einen bestimmten Endopeptidasespaltungstyp Sequenzinformation liefern

<i>Fragmente, die</i>	LysC	GluC	Trypsin	Chymotrypsin	AspN	Mittelwert über alle Spaltungen
<i>N-terminale Sequenzierung</i>						
keine Sequenzinformation liefern	58%	35%	41%	66%	61%	52%
Sequenzinformation liefern	42%	65%	59%	34%	39%	48%
<i>C-terminale Sequenzierung</i>						
keine Sequenzinformation liefern	29%	41%	71%	31%	50%	44%
Sequenzinformation liefern	71%	59%	29%	69%	50%	56%

Mittelwerte über N- und C-terminale Sequenzierung:

Fragmente, die keine Sequenzinformation liefern: 48%

Fragmente, die Sequenzinformation liefern: 52%

Die Verteilung des Auftretens bestimmter Sequenzlängen nach den unterschiedlichen Endopeptidasespaltungen ist in den folgenden vier Grafiken zusammengestellt. Da für Endoproteinase AspN mit *DrTI* (*Serin Proteinase Inhibitor*) nur ein Protein untersucht wurde, stehen hier nicht genügend Daten über Sequenzlängen zur Verfügung. Spaltungen mit Endoproteinase AspN wurden daher in den folgenden Grafiken nicht berücksichtigt.

Es ist festzustellen, das für alle Endopeptidasespaltungen, sowohl die N- als auch die C-terminale Sequenzierung nur in Einzelfällen Teilsequenzen mit mehr als acht Aminosäuren eines Peptids liefert. Die am häufigsten auftretenden Sequenzen besitzen bei N-terminalem, ebenso wie bei C-terminalem Abbau eine Länge zwischen ein und sechs Aminosäuren. Die Verteilungen zeigen im Mittel für alle Endopeptidasespaltungen eine Abnahme der Häufigkeit ab Teilsequenzen mit mehr als drei bis fünf Aminosäuren. Dies wird bei einer Betrachtung von Abbildung 36 und Abbildung 37 deutlich. Abbildung 36 stellt die gemittelten Resultate

aller Endopeptidasespaltungen für N- und C-terminale Sequenzierungen gegenüber, während in Abbildung 37 der Durchschnittswert aus der N- und C-terminalen Sequenzdaten für alle Endopeptidasen aufgetragen wurde.

Abbildung 34: Verteilung N-terminaler Sequenzlängen nach verschiedenen Endopeptidasespaltungen

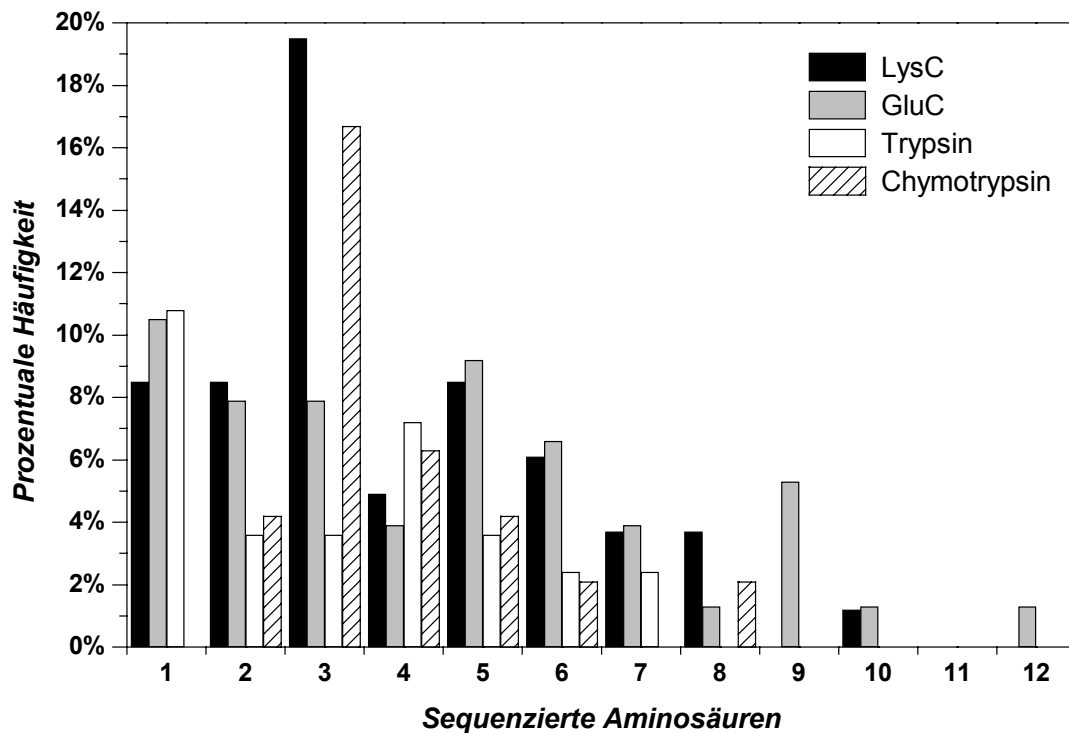


Abbildung 35: Verteilung C-terminaler Sequenzlängen nach verschiedenen Endopeptidasespaltungen

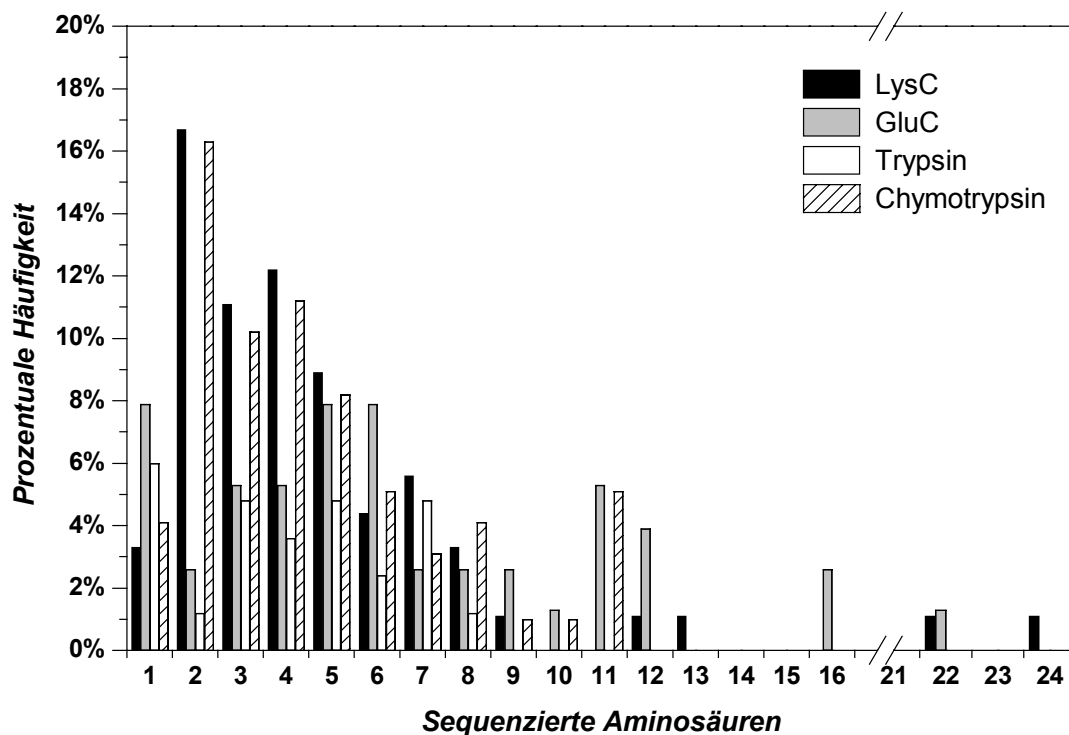


Abbildung 36: Häufigkeiten von N- und C-terminalen Sequenzlängen im Vergleich, gemittelt über die unterschiedlichen Endopeptidasespaltungen

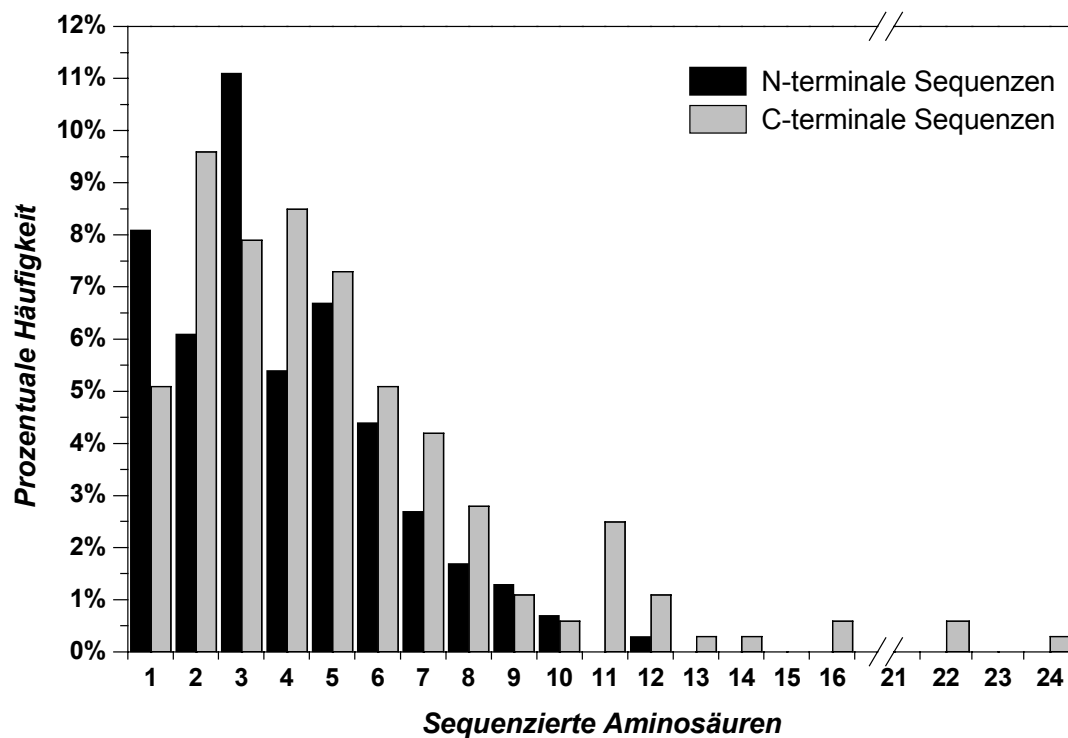
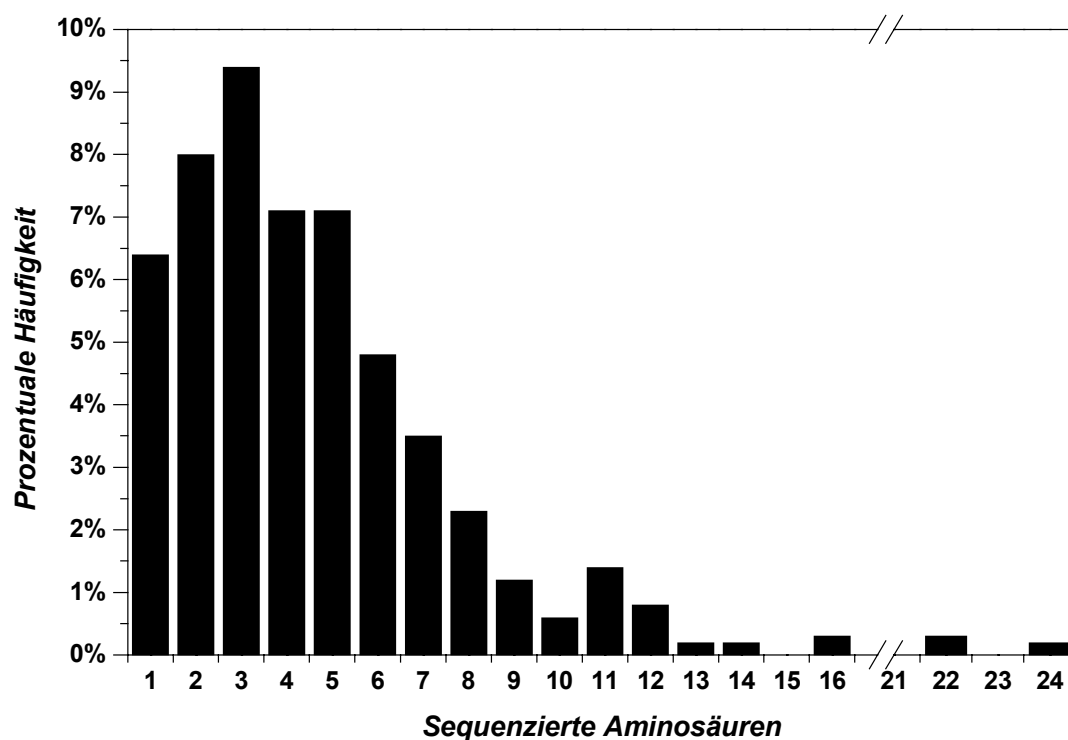
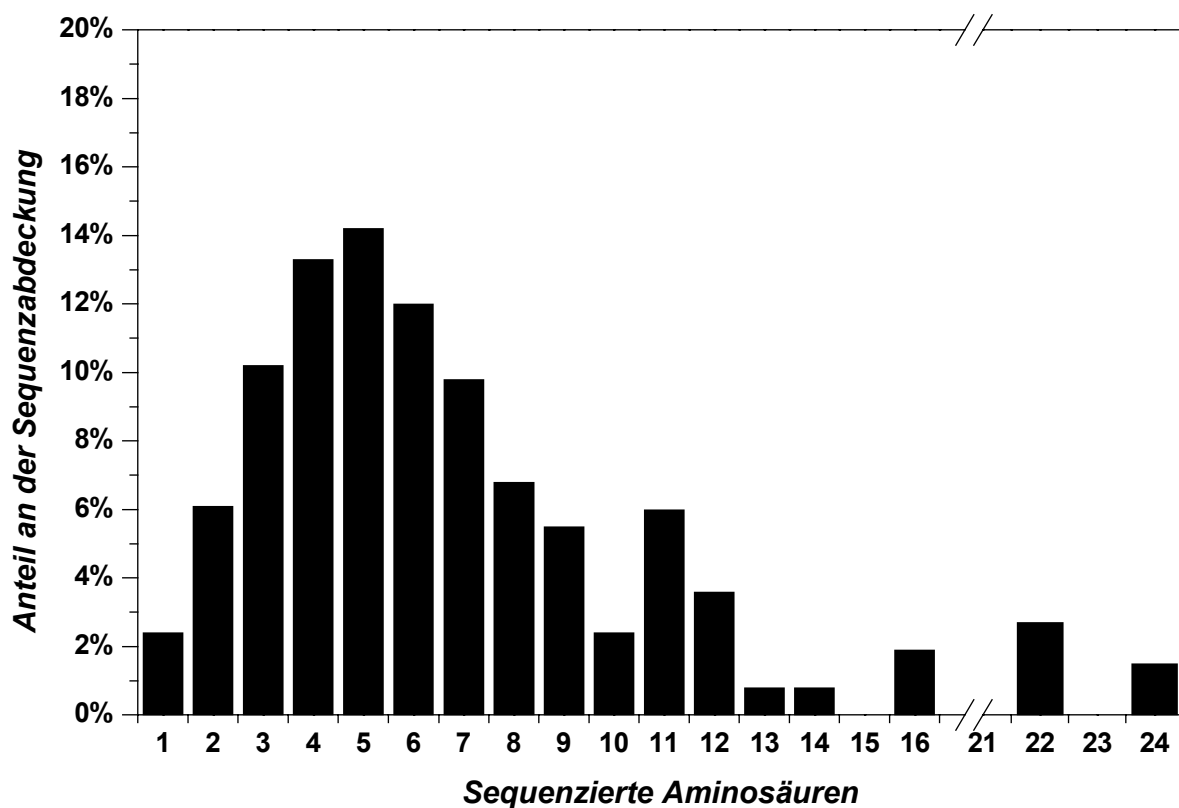


Abbildung 37: Gemittelte Häufigkeiten N- und C-terminaler Sequenzlängen, gemittelt über alle Endopeptidasespaltungen



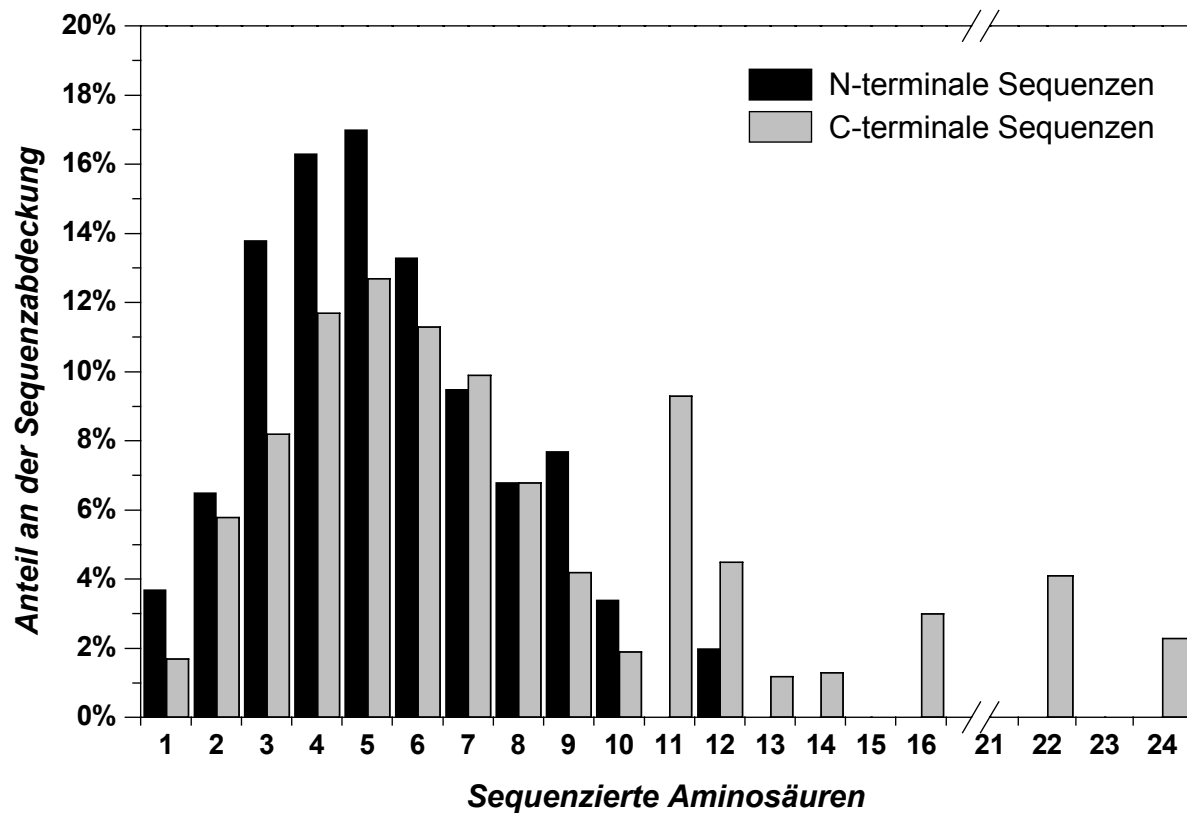
Insgesamt wurden 1650 Aminosäuren sequenziert, davon 588 durch N-terminalen und 1062 C-terminalen Abbau. Bezieht man nun die durch eine bestimmte Sequenzlänge abgedeckte Zahl an Aminosäuren auf diese Werte, so lässt sich die Bedeutung einer auftretenden Sequenzlänge für die Sequenzabdeckung abschätzen. Bezogen auf alle sequenzierten Aminosäuren (entspricht 100%) zeigt Abbildung 38, dass der größte Teil der erhaltenen Sequenzabdeckung aus Teilsequenzen mit vier bis sechs Aminosäuren stammt. Sequenzen aus der Spaltung mit Endoproteinase AspN wurden auch hier, wie bereits oben begründet, nicht in die Beurteilung miteinbezogen.

Abbildung 38: Anteile verschiedener Sequenzlängen an der Sequenzabdeckung, gemittelt über alle Endopeptidasespaltungen



Die Verteilung zeigt ein nahezu gaußförmiges Profil. Sehr lange Sequenzen können demnach auch bei geringerer Häufigkeit einen hohen Anteil an der Sequenzabdeckung ausmachen. Dies zeigt sich insbesondere sehr deutlich bei einer getrennten Betrachtung der Resultate für N- und C-terminaler Sequenzierung. Zwar treten C-terminale Sequenzen mit 11 Aminosäuren nur mit einer Häufigkeit von 2,5% auf, allerdings machen diese langen Sequenzen 8,1% der C-terminal, und somit immerhin 5,9% der gesamten, erreichten Sequenzabdeckung aus (siehe Abbildung 39 und Abbildung 38).

Abbildung 39: Anteile verschiedener Sequenzlängen an der Sequenzabdeckung, gemittelt über alle Endopeptidasespaltungen – getrennte Betrachtung nach N- und C-terminaler Sequenzierung



Bei einzelner Betrachtung der Ergebnisse, die sich für die verschiedenen untersuchten Endopeptidasespaltungen ergeben (Abbildung 40 und Abbildung 41), erscheint das Bild zunächst wesentlich uneinheitlicher zu sein. Tatsächlich ergeben sich jedoch auch hier weitgehend gaußförmige Verteilungen.

Betrachtet man zunächst die N-terminale Sequenzierung (Abbildung 40), so trifft dies insbesondere für die Spaltungen mit Endoproteinase LysC und Chymotrypsin zu. Diese haben ein Maximum bei Sequenzen mit vier (Endoproteinase LysC) respektive drei (Chymotrypsin) Aminosäuren. Für Spaltungen mit Trypsin, aber auch Endoproteinase GluC (Phosphatpuffer), kann allerdings ebenfalls von einem weitgehend gaußförmigen Profil mit Maxima bei Sequenzen mit vier (Trypsin) beziehungsweise fünf (Endoproteinase GluC) Aminosäuren gesprochen werden. Ausreißer sind bei Sequenzen mit sieben (Trypsin) beziehungsweise neun Aminosäuren (Endoproteinase GluC) zu beobachten. Abbildung 34 zeigt, dass diese Sequenzlängen bei Spaltungen mit Trypsin beziehungsweise Endoproteinase GluC in Relation zu anderen Sequenzlängen sehr häufig aufgetreten sind.

Abbildung 40: Anteile verschiedener Sequenzlängen an der Sequenzabdeckung – getrennte Betrachtung für unterschiedliche Endopeptidasespaltungen bei N-terminalen Sequenzen

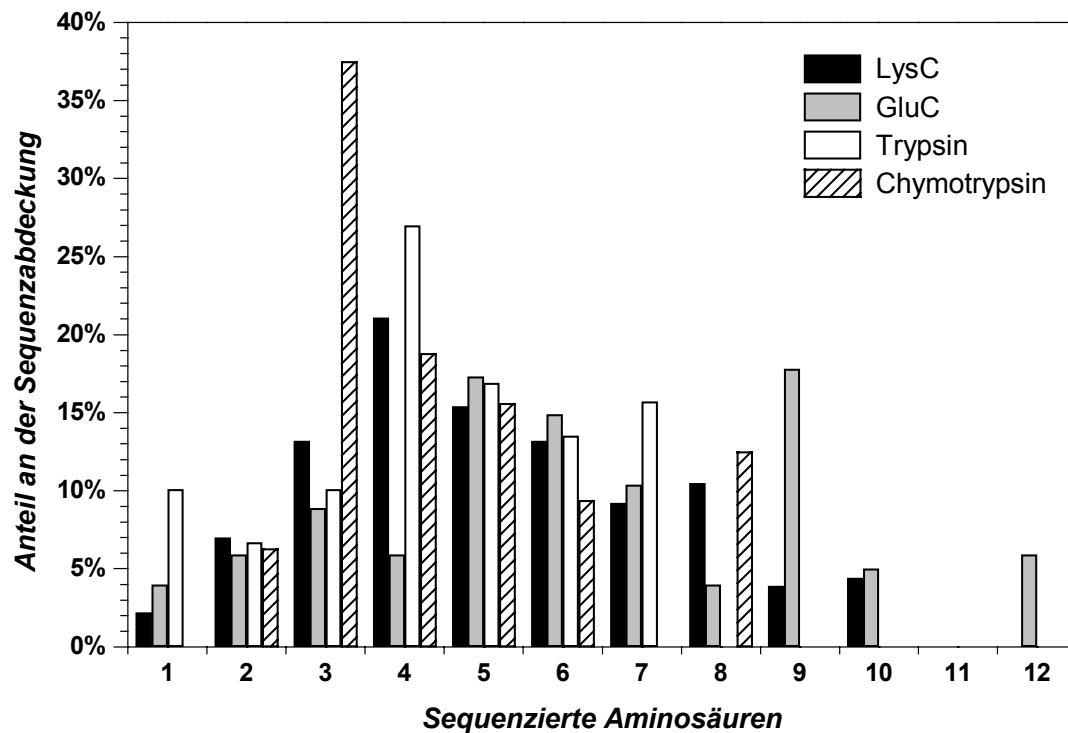
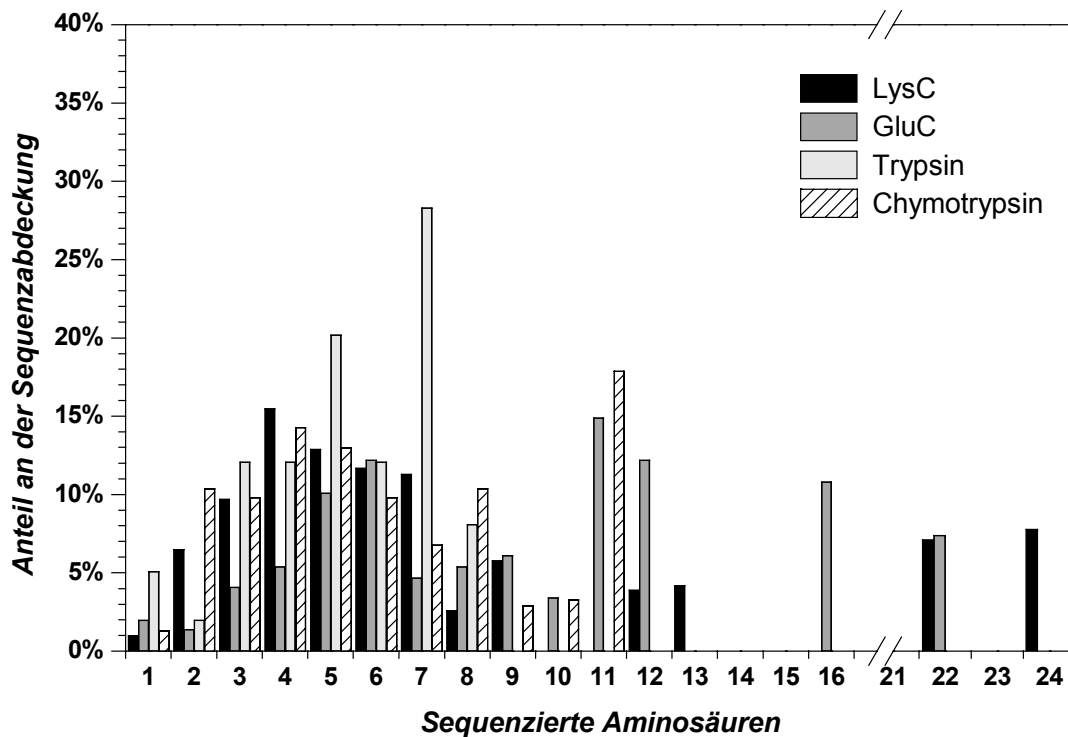


Abbildung 41: Anteile verschiedener Sequenzlängen an der Sequenzabdeckung – getrennte Betrachtung für unterschiedliche Endopeptidasespaltungen bei C-terminalen Sequenzen



Bei den Resultaten C-terminaler Sequenzierungen (Abbildung 41) erscheinen die gaußförmigen Profile weniger stark ausgeprägt. Für Sequenzlängen bis 10 Aminosäuren und deren Anteile an der erhaltenen Sequenzabdeckung sind diese Profile jedoch ebenfalls größtenteils gegeben (mit der Ausnahme einer starken Abweichung bei Sequenzen mit sieben Aminosäuren für Spaltungen mit Trypsin). Längere Sequenzen als 10 Aminosäuren sind im Gegensatz zur N-terminalen Sequenzierung (0,3%) bei C-terminaler Sequenzierung häufiger anzutreffen, nämlich bei 5,6% aller sequenzierten Peptide. Im Mittel (über alle Endopeptidasespaltungen) decken solche langen Sequenzen bei der C-terminalen Sequenzierung immerhin ein Viertel der gesamt erzielten Sequenzinformation ab. Abbildung 40 verdeutlicht, dass der Anteil, den Sequenzen mit mehr als 10 Aminosäuren bei einzelnen Endopeptidasespaltungen an der erhaltenen Sequenzabdeckung haben zum Teil sehr groß sein kann. So beträgt er beispielsweise für Spaltungen mit Chymotrypsin 18%, für Spaltungen mit Endoproteinase LysC 23% und für Spaltungen mit Endoproteinase GluC (Phosphatpuffer) sogar 45% !

3.2.3.4 Sequenzlängenverteilung in Abhängigkeit der Exopeptidasespaltung

Die folgenden Tabellen geben einen Überblick über die Anteile sequenzliefernder Fragmente bei Anwendung bestimmten Exopeptidasespaltungsgruppen (vergleiche Tabelle 15 bis Tabelle 18). Die angegebenen Werte für eine Exopeptidasespaltungsgruppe umfassen die Resultate aus allen Endopeptidasespaltungen.

N-terminale Sequenzierung:

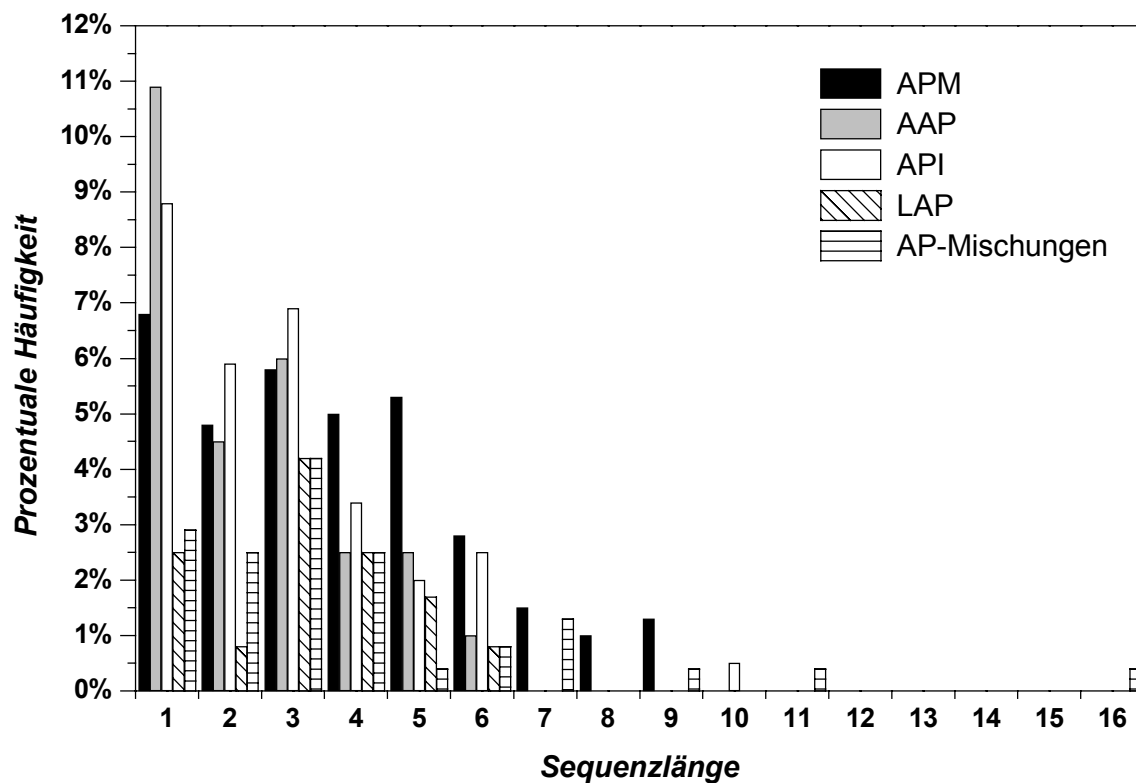
Tabelle 34: Anteile der gefundenen Spaltfragmente, die für eine bestimmten Exopeptidasespaltungstyp Sequenzinformation liefern (N-terminal) – differenziert für APM-Sequenzierungen

Fragmente, die	APM	AAP	API	LAP	multi(AP)
keine Sequenzinformation liefern	66%	73%	70%	87%	84%
Sequenzinformation liefern	34%	27%	30%	13%	16%

Fragmente, die	APM ₁	APM ₂
keine Sequenzinformation liefern	60%	84%
Sequenzinformation liefern	40%	16%

Bei N-terminaler Sequenzierung liefen auf den ersten Blick nur die Einzelpeptidasen APM, AAP und API größere Anteile der Sequenzinformation. Die Werte geben jedoch zunächst keine Auskunft über die Qualität der erhaltenen Information, falls sich ein Peptid mit einer bestimmten Peptidase oder Peptidasekombination sequenzieren lässt. Zwar erhält man mit verschiedenen AP-Kombinationen „multi(AP)“ gemäß Tabelle 16 nur in 16% der Fälle eine Sequenz, allerdings sind diese Sequenzen dann auch meist wesentlich länger oder beziehen sich auf einen Sequenzabschnitt, der weiter vom ursprünglichen N-Terminus entfernt ist. Abbildung 42 zeigt, dass im wesentlichen nur für Aminopeptidase-Mischungen einige längere Sequenzen erhalten werden konnten.

Abbildung 42: Verteilung N-terminaler Sequenzlängen bei verschiedenen Exopeptidasespaltungen (Einzelpeptidasen und Peptidasekombinationen)



Die besten Resultate als Einzelpeptidase liefert die microsomale Aminopeptidase **APM**. Insbesondere bei vergleichbarer Reaktionszeit (1-6min) gegenüber AAP und API sind die Chancen auf eine mögliche Sequenzierung eines getesteten Peptids mit 40% signifikant höher als bei AAP oder API. Im Bezug auf die Verteilung der Häufigkeit bestimmter Sequenzlängen im Falle eines erfolgreichen Sequenzierungsversuchs verhalten sich die drei letztgenannten Aminopeptidasen bei kurzen Sequenzen (bis zu vier Aminosäuren) sehr ähnlich. Längere Sequenzen (fünf bis neun) Aminosäuren treten jedoch mit APM wesentlich häufiger auf

beziehungsweise lassen sich für AAP oder API überhaupt nicht beobachten. Auch LAP und AP-Mischungen zeigen qualitativ eine zu den anderen drei anderen Peptidasen vergleichbare Verteilung der Sequenzlängen. Da für LAP und AP-Mischungen allerdings in deutlich weniger Fällen überhaupt eine Sequenz erhalten wurde (13% bzw. 16%) liegen die prozentualen Häufigkeiten der einzelnen Sequenzlängen gleichermaßen wesentlich niedriger.

C-terminale Sequenzierung:

Tabelle 35: Anteile der gefundenen Spaltfragmente, die für eine bestimmten Exopeptidasespaltungstyp Sequenzinformation liefern (C-terminal) – differenziert für CPY- und CPP-Sequenzierungen

<i>Fragmente, die</i>	CPY	CPP	CPA	CPW	sb(CP-I)	sb(CP-II)	sb(CP-III)	pb(CP)	spt(CP)
keine Sequenzinformation liefern	65%	78%	91%	78%	58%	79%	91%	67%	91%
Sequenzinformation liefern	35%	22%	9%	22%	42%	21%	9%	33%	9%

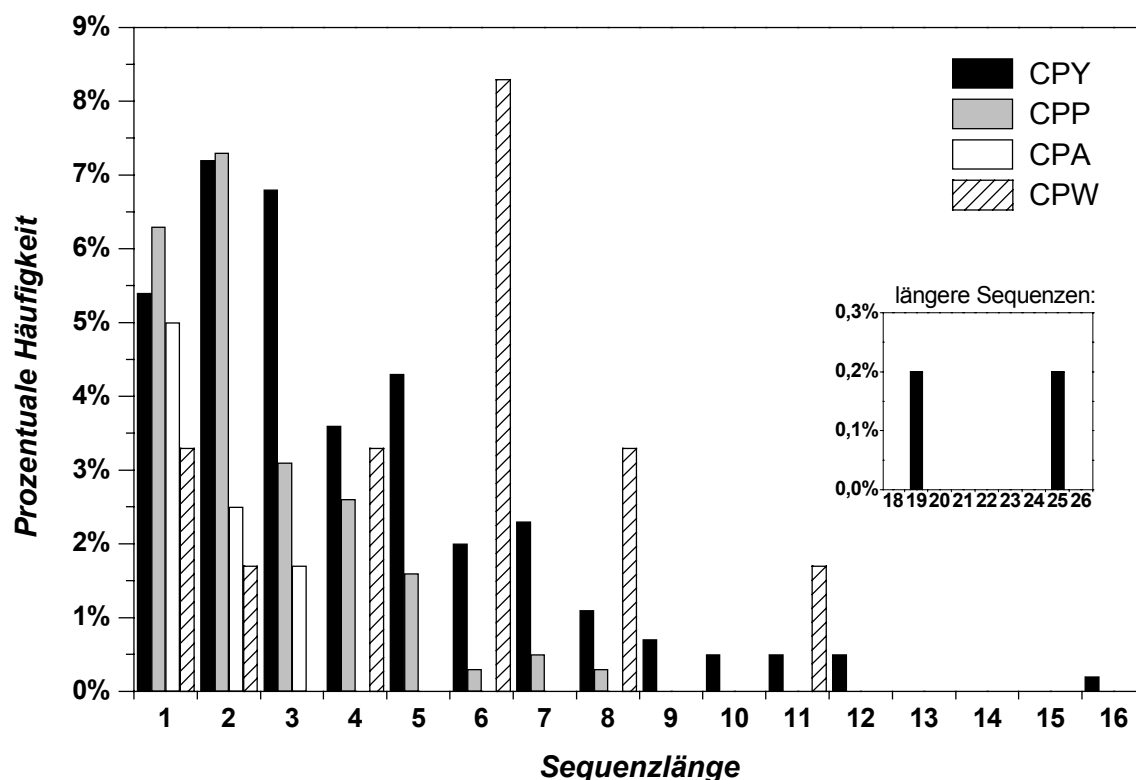
<i>Fragmente, die</i>	CPY₁	CPY₂	CPP₁	CPP₂
keine Sequenzinformation liefern	51%	88%	67%	95%
Sequenzinformation liefern	49%	12%	33%	5%

Bei C-terminaler Sequenzierung sind die Aussichten Sequenzinformation zu erhalten für die Gruppen **CPY**, **sb(CP-I)** und **pb(CP)** am größten. Wie Tabelle 18 zeigt enthalten die beiden dabei betroffenen Peptidasekombinationen ebenfalls nur CPY und CPP. Gute Werte ergeben sich auch für die Einzelpeptidasen CPP, CPW und die seriellen Kombinationen aus CPB und einer unspezifischen Peptidase.

Bei der grafischen Darstellung der prozentualen Häufigkeitsverteilungen bestimmter Sequenzlängen, wurde im Fall der C-terminalen Sequenzierungen zur besseren Übersicht nach Einzelpeptidasen und Peptidasekombinationen differenziert. Die Verteilung der Häufigkeiten bei den getesteten Einzelpeptidasen erscheint sehr unregelmäßig (Abbildung 43). Der Schwerpunkt liegt für alle Peptidasen bei kurzen Sequenzen mit bis zu 5 Aminosäuren. Insbesondere bei CPY treten allerdings auch längere Sequenzen mit bis zu 7 – 8 Aminosäuren noch in einem nennenswerten Umfang auf. Sehr lange Sequenzen mit bis zu maximal 25 Aminosäuren konnten zwar für CPY auch beobachtet werden, bilden jedoch die Ausnahme. Die Zahl der mit CPW und CPA sequenzierten Peptide war im Vergleich zu den anderen Peptidasen eher gering (siehe auch Tabelle 19). Die Häufigkeit des Auftretens bestimmter Sequenzlängen

bei diesen beiden Peptidasen ist daher statistisch weniger gut abgesichert. Bei der Sequenzierung mit CPA ergaben jedoch alle Eingangsversuche eher schlechte Resultate, die zumeist weit hinter den Ergebnissen von CPY und CPP zurückblieben. Die nur 9% aller Fälle, mit denen bei der Anwendung von CPA Sequenzinformation erhalten wurde, verteilen sich zudem auch noch auf sehr kurze Sequenzen mit maximal 3 Aminosäuren. Daher erfolgte eine eingehendere, systematischere Untersuchung mit CPA nicht. Anders verhält es sich mit CPW: diese Peptidase wurde erst im spätern Verlauf der Arbeit verfügbar. Die Darstellung in Abbildung 43 für CPW ist daher lückenhaft. Es sollte zwar nach den bisherigen Versuchen noch nicht davon ausgegangen werden, dass das Maximum der Häufigkeitsverteilung bei Sequenzen einer Länge um 6 Aminosäuren liegt. Allerdings deuten die Resultate für CPW auf ein gutes Potential für die Leitersequenzierung hin.

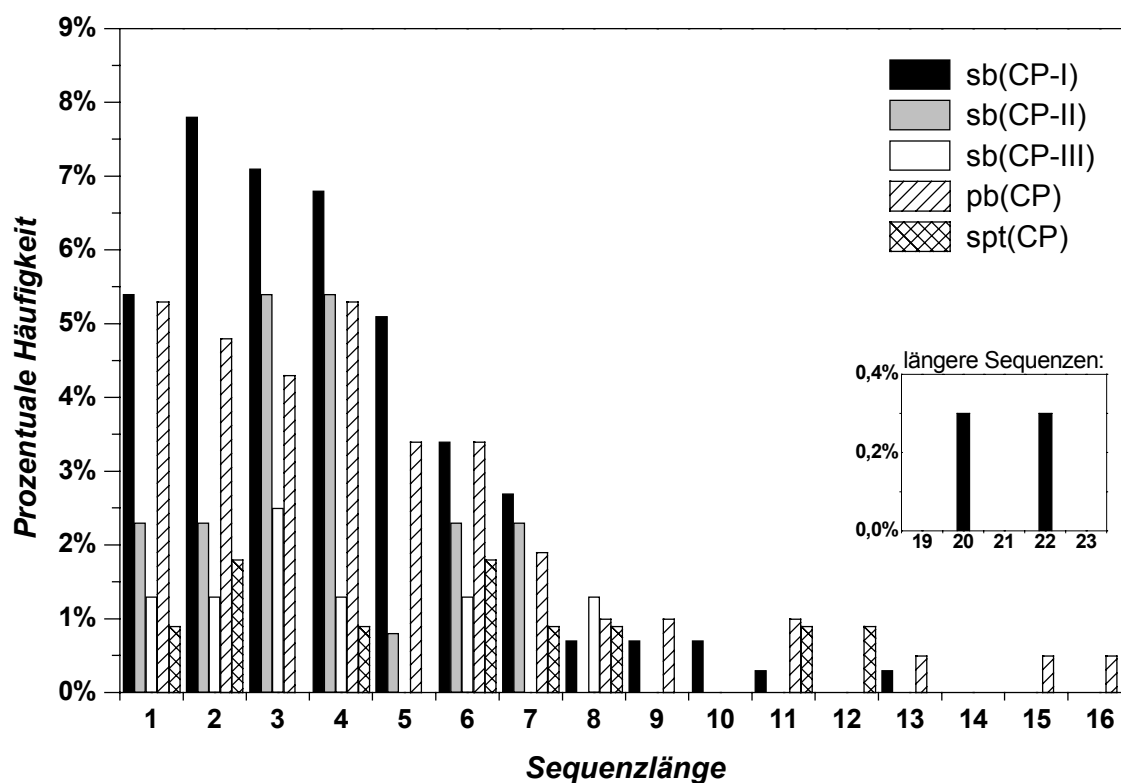
Abbildung 43: Verteilung C-terminaler Sequenzlängen bei verschiedenen Exopeptidasespaltungen (Einzelpeptidasen)



Bei Carboxypeptidasekombinationen sind längere Sequenzen mit bis zu 7 Aminosäuren häufiger zu beobachten als bei der Sequenzierung mit einzelnen Exopeptidasen. Dies trifft besonders für die Kombinationen sb(CP-I) und pb(CP) zu, deren Anteile sequenzliefernder Peptide mit 43% respektive 33% zudem noch sehr hoch liegt. Die ternären Mischungen verschiedener Carboxypeptidasen liefern zum Teil längere Sequenzen, allerdings ist zum einen

der sequenzliefernde Anteil mit 9% sehr niedrig, zum anderen kann beim direkten Vergleich der Sequenzierung mit einer ternären Mischung zu einer Sequenzierung mit binärer Mischung meist keine entscheidende Verbesserung (systematisch längere Sequenzen oder zusätzliche Information) festgestellt werden. Ebenso zeigen binäre Mischungen anderer unspezifischer Peptidasen (CPY + CPA, CPY + CPW oder CPA + CPP) in sb(CP-III) keine vergleichbar guten Ergebnisse wie die binären Kombinationen aus CPY und CPP in sb(CP-I). Allerdings bedürfen auch in diesem Fall Kombinationen aus CPY oder CPP mit CPW noch einer weiterführenden Untersuchung. Ebenso wie bei den Einzelpeptidasen erhält man auch mit Peptidasekombinationen nur in einigen wenigen Einzelfällen sehr lange Sequenzen (über 15 Aminosäuren).

Abbildung 44: Verteilung C-terminaler Sequenzlängen bei verschiedenen Exopeptidasespaltungen (Peptidasekombinationen)



Spezielle Betrachtung der Sequenzierung mit der Peptidasekombination sb(CP-II):

Die Peptidasekombination sb(CP-II) wurde nur für Fragmente aus den Spaltungen mit den Endoproteinasen LysC und Trypsin verwendet. Der gute Wert für sequenzliefernde Peptide bei sb(CP-II) ist also mit Einschränkung zu betrachten. Aufgrund der Spezifität von CPB für die C-terminale Abspaltung von Lys und Arg sind die Resultate hier natürlich

überproportional gut. Im folgenden ist dies am Beispiel der Spaltung des Cytochrom C mit Endoproteinase LysC dargestellt. Tabelle 36 zeigt die bei der Spaltung theoretisch zu erwartenden Fragmente (für MC = 0), sowie die Wiederfindung und Sequenzierbarkeit dieser Fragmente. Der sequenzierbare Anteil ($m > 700\text{Da}$) ist grau unterlegt.

Tabelle 36: Theoretische Fragmente einer Spaltung von Cytochrom C mit Endoproteinase LysC (MC = 0), praktische Wiederfindung nach HPLC und Sequenzierbarkeit

Sequenz	Position	Masse $[M+H]^+$	HPLC	theoretisch sequenzierbar
EETLMEYLENPK	61- 72	1495.70	+	+
TGQAPGFTYTDANK	40- 53	1470.69	+	+
TEREDLIAYLK	89- 99	1350.73	+	+
TGPNLHGLFGRK	28- 39	1269.72	+	+
CAQCHTVEK	14- 22	1018.45	+ ¹	+
MIFAGIK	80- 86	779.45	+	+
YIPGTK	74- 79	678.38	+	-
IFVQK	9- 13	634.39	+	-
GITWK	56- 60	604.35	-	-
GDVEK	1- 5	547.27	-	-
ATNE	101-104	414.19	-	-
HK	26- 27	284.17	-	-
NK	54- 55	261.16	-	-
GGK	23- 25	261.16	-	-
GK	6- 7	204.13	-	-
K	100-100	147.11	-	-
K	88- 88	147.11	-	-
K	87- 87	147.11	-	-
K	73- 73	147.11	-	-
K	8- 8	147.11	-	-

¹ Das Fragment 14-22 findet sich nur im unvollständig gespaltenen Fragment IFVQKCAQCHTVEK (9-22)

Bezogen auf das gesamte Protein mit 104 Aminosäuren (entspricht 100%) beträgt der wiedergefundene Anteil des Proteins nach HPLC 73% (76 AS) und der sequenzierbare Anteil 63% (65 AS). In Tabelle 37 und Tabelle 38 sind die Resultate der Sequenzierungen dieser 63% mit der Einzelpeptidase CPY und der Kombination sb(CP-II) (CPB + CPY) einander gegenübergestellt.

Tabelle 37: Endoproteinase LysC-Fragmente von Cytochrom C – Sequenzierung mit CPY

Sequenz	Position	Masse [M+H] ⁺	sequenzierte AS
EETLMEYLENPK	61- 72	1495.70	2
TGQAPGFTYTDANK	40- 53	1470.69	3
TEREDLIAYLK	89- 99	1350.73	3
TGPNLHGLFGRK	28- 39	1269.72	2
CAQCHTVEK	14- 22	1018.45	3
MIFAGIK	80- 86	779.45	0

Tabelle 38: Endoproteinase LysC-Fragmente von Cytochrom C – kombinierte Sequenzierung mit CPB und CPY

Sequenz	Position	Masse [M+H] ⁺	sequenzierte AS
EETLMEYLENPK	61-72	1495.70	7
TGQAPGFTYTDANK	40-53	1470.69	6
TEREDLIAYLK	89-99	1350.73	5
TGPNLHGLFGRK	28-39	1269.72	5
CAQCHTVEK	14-22	1018.45	4
MIFAGIK	80-86	779.45	0

In der Zusammenfassung ergibt sich damit im Vergleich zur Einzelpeptidase eine doppelt so hohe Sequenzabdeckung für die Peptidasekombination sb(CP-II):

Einzelpeptidase CPY:

Sequenzabdeckung bezogen auf die Gesamtsequenz (104 AS): **13%**

Sequenzabdeckung bezogen auf den sequenzierbaren Anteil (65 AS): **20%**

Peptidasekombination sb(CP-II) (CPB + CPY):

Sequenzabdeckung bezogen auf die Gesamtsequenz (104 AS): **26%**

Sequenzabdeckung bezogen auf den sequenzierbaren Anteil (65 AS): **42%**

Für die sequenzliefernden Peptide steigt die mittlere Sequenzlänge von 2-3 Aminosäuren bei alleiniger Sequenzierung mit CPY auf 5-6 Aminosäuren bei Anwendung von sb(CP-II). Dies entspricht einer Zunahme der Sequenzinformation um 100-200% bei Anwendung der sequenziellen Peptidasekombination gegenüber CPY.

3.3 Derivatisierte Peptide

Für eine Derivatisierung zur Verbesserung einer enzymatischen Leitersequenzierungsmethode sprechen mehrere Gründe, die bereits im Theorieteil unter Punkt 1.4.3 (S. 61) aufgezählt und erläutert wurden. Die Derivatisierungen in dieser Arbeit wurden vor allem mit zweierlei Zielen durchgeführt:

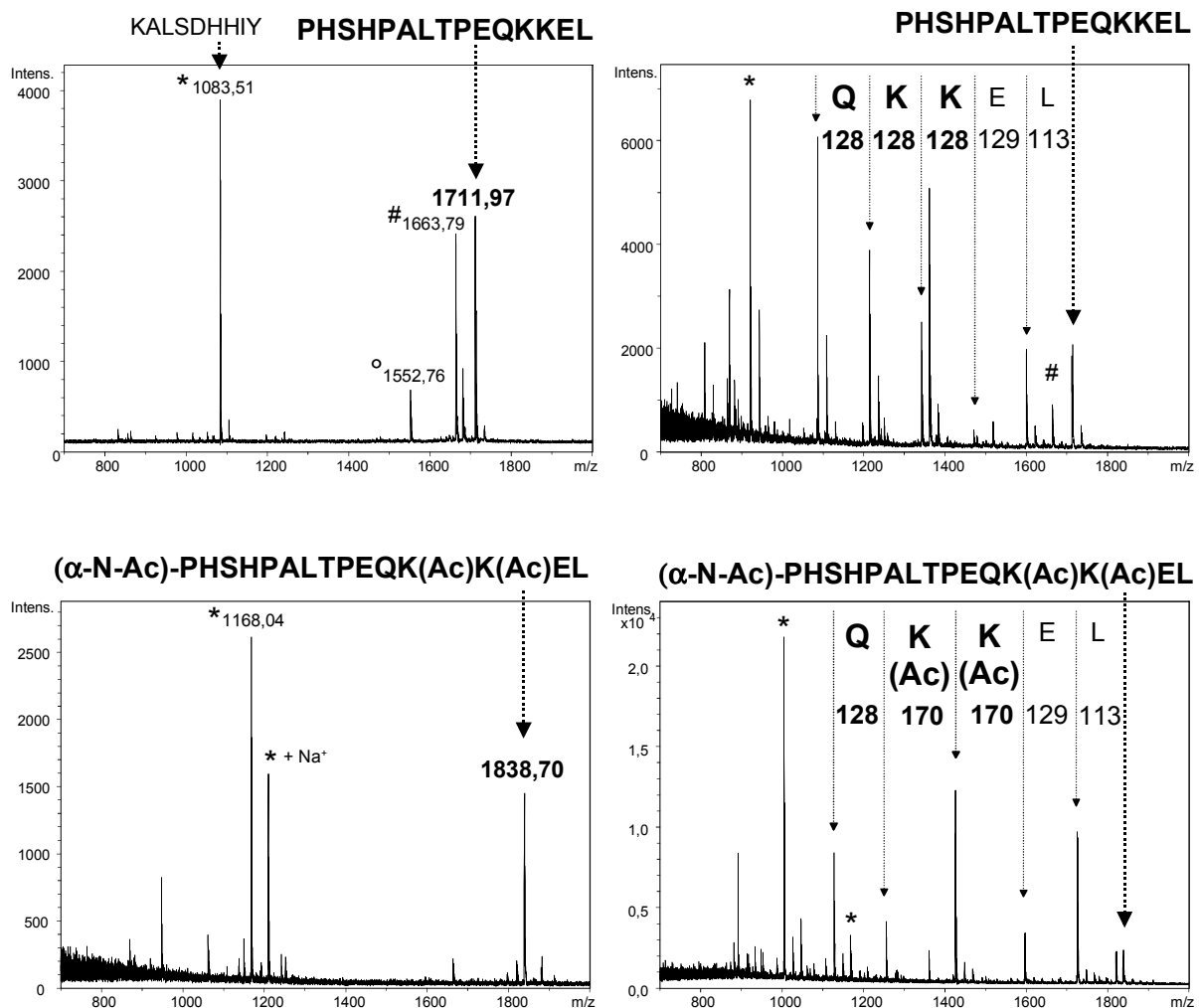
- ❖ Durch eine selektive Derivatisierung sollte eine einfache Unterscheidung zwischen Lysin und Glutamin ermöglicht werden. Deren Massendifferenz erlaubt bei der gegebenen Massengenauigkeit und Präzision keine direkte Unterscheidung der nicht-modifizierten Aminosäuren über MALDI-MS.
- ❖ Die durch die Derivatisierung eingeführte Gruppe sollte eine Massenerhöhung für das Peptid zur Folge haben. Zielsetzung war dabei, zusätzlich die Sequenzierung kleinerer Peptide zu ermöglichen und damit die Sequenzabdeckung bei einfacher Sequenzierung der Proteinfragmente von nur einem Terminus her erhöhen.

3.3.1 Derivatisierung zur Unterscheidung Glutamin / Lysin

Eine Derivatisierung mit dem Ziel der Unterscheidung Gln/Lys erfordert vom Derivatisierungsprodukt, dass sich die derivatisierte Aminosäure enzymatisch auch weiterhin abbauen lässt. Die ohnehin für alle Proteine durchgeführte Behandlung des Cysteins mit Vinylpyridin hat bereits gezeigt, dass ein Abbau derivatisierter Aminosäuren durch Exopeptidasen grundsätzlich möglich ist. Laut [Bonetto 1997] ermöglicht eine solche Derivatisierung des Cysteins überhaupt erst dessen C-terminalen Abbau. Im Gegenzug lassen sich nach den Autoren weder underivatisiertes Cystein, noch Cysteinsäure oder carboxymethyliertes Cystein durch Carboxypeptidasen abbauen. In einem kombinierten, chemisch-enzymatischen Abbaubersuch gelang Thiede, Salnikow und Wittmann-Liebold [Thiede 1997] eine Unterscheidung des Gln/Lys durch das in einer Nebenreaktion gleichzeitig acetylierte Lysin. Das von diesen Autoren dargestellte Verfahren erscheint jedoch für die reine Unterscheidung Gln/Lys in dieser Durchführungsform ungeeignet – vor allem aufgrund der langen Reaktionsdauer von 6-16 Stunden. Auch Bonetto *et al.* beschreiben zur Unterscheidung Lys/Gln eine effiziente Guanidinylierung des Lysins mittels O-Methylisoharnstoff zu Homoarginin [Bonetto 1997]. Auch diese Umsetzung dauert aber mit 4-5 Stunden Reaktionszeit sehr lang und erfordert zudem eine Aufreinigung des Derivats mittels C-18 RP-HPLC vor der Sequenzierung.

Im geforderten Kontext für die enzymatische Leitersequenzierung sollte die Derivatisierung gezielt, kontrolliert, aber auch schnell erfolgen. Erfolgreich konnten diese Forderungen durch eine Acetylierung mittels Essigsäureanhydrid/Eisessig in Aceton, wie unter 2.7.4 beschrieben, realisiert werden. Die Reaktionsdauer ist mit 60min bei Raumtemperatur (27°C) im Vergleich zu den anderen beschriebenen Verfahren sehr kurz. Eine Erhöhung der Reaktionstemperatur zur Verkürzung der Reaktionszeit ist nicht ratsam. Die Acetylierungsversuche zeigten bei höheren Temperaturen eine zu starke Zerstörung des Peptids. Die anschließende C-terminale Sequenzierung des derivatisierten Lysins stellt kein Problem dar. Eine Aufreinigung (Entsalzung) des Derivats vor der Sequenzierung ist ebenfalls nicht notwendig, da sich das überschüssige Reagenz (Essigsäure) bei der Vakuumzentrifugation vollständig verflüchtigt. Abbildung 45 zeigt ein Resultat für das Fragment 1-15 aus einer Spaltung der Aldolase mit Chymotrypsin.

Abbildung 45:: Vergleich der C-terminale Sequenzierung eines nicht-acetylierten und eines acetylierten Peptids aus einer Proteinspaltung der Aldolase mit Chymotrypsin (° / # : unbekannte Substanzen)



Gegenübergestellt sind die Resultate für das underivatisierte Peptid (oben) und das derivatisierte Peptid (unten). Auf der linken Seite sind die Ausgangspeptide vor der Sequenzierung zu sehen, rechts die dazugehörigen Leiterpeptidspektren. Die Sequenzierung erfolgte in beiden Fällen mit sb(CP-I) (serielle Kombination aus CPY+CPP). Das Peptid enthält zwei Lysine und den freien N-Terminus als potenzielle Derivatisierungsstellen. Diese basischen funktionellen Gruppen wurden bei der Acetylierung komplett derivatisiert – der gefundene *Massenshift* entspricht einer dreifachen Acetylierung. Die ebenfalls basischen Histidine wurden unter gewählten Reaktionsbedingungen nicht angegriffen.

Das rechte untere Leiterspektrum des Derivats zeigt, dass eine eindeutige Unterscheidung des Lysins zum Glutamin möglich ist. Auch das starke Nebensignal der gewählten Fraktion bei ca. 1083 Da (markiert mit Stern *; Fragment 214-222 aus der Spaltung von Aldolase mit Chymotrypsin; Sequenz KALSDHHIY) wird zweifach acetyliert (Derivatmasse ca. 1168 Da). Die Acetylierung findet hier ebenfalls nur am N-Terminus und am Lysin statt. Das Sequenzierungsergebnis für das Nebensignal beweist für dieses Peptid, dass keine O-acetylierung am Tyrosin stattfindet, denn der mit Stern (*) gekennzeichnete Peak im unteren rechten Spektrum weist zum Signal bei 1168 Da eine Massendifferenz von 163 Da für ein underivatisiertes Tyrosin auf. Ein N-terminaler Abbau nach einer solchen Acetylierung war nicht mehr möglich. Die Acetylierung des N-Terminus blockiert die N-terminale Sequenzierung offensichtlich effektiv. Auch die nachfolgend gezeigten Derivatisierungen machen deutlich, dass nach einer Umsetzung des N-Terminus Sequenzierungen auf die C-terminale Seite beschränkt sind. Im Fall der Acetylierung besteht jedoch der Vorteil die N-terminale Blockierung durch Anwendung von Acylaminoacyl-Peptidase (EC 3.4.19.1) wieder aufzuheben.

3.3.2 Derivatisierung zur Sequenzierung kleiner Peptide

3.3.2.1 Überlegungen zur Auswahl von Derivatisierungsreagenzien

Ein Blick auf Tabelle 31 zeigt, dass der Anteil von Peptidfragmenten mit einer Masse unter 700 Da nach Endopeptidasespaltung im Mittel bei ca. 20% liegt. Zudem wurde bereits unter 1.2.2.3 erörtert, dass bei einer nur einseitigen Sequenzierung (nur N-terminal oder nur C-terminal) die letzten 3, oftmals sogar die letzten 4-5 Aminosäuren des Terminus, welcher der Sequenzierungsseite gegenüberliegt, auf keinen Fall durch die Leitersequenzierung analytisch fassbar sind (Überlappung mit dem Matrixbereich).

Daher wurde versucht, über geeignete Derivatisierungsreagenzien einen Massenshift einzuführen, der alle Leiterpeptide in ihrer Masse weitgehend aus dem Bereich der Massen der Matrixionen „herausschiebt“. Praktikabel erscheinen hier zwei Vorgehensweisen:

- ❖ Derivatisierung eines Peptidterminus
- ❖ Derivatisierung der kompletten Leiter nach Sequenzierung

Im Rahmen dieser Arbeit wurde die erste Variante eingehender untersucht. Aufgrund der Reaktivitäten bietet hier, ohne vorherige Aktivierung, nur die Umsetzung des N-Terminus eine Derivatisierungsmöglichkeit. Daher wurden Reagenzien ausgewählt, die eine möglichst vollständige Umsetzung des N-Terminus bei gleichzeitig hoher Massenverschiebung erlauben. Gleichzeitig wurde bei der Auswahl der Reagenzien jedoch auch Wert darauf gelegt, möglichst viele der unter 1.4.3 aufgelisteten, positiven Eigenschaften einer Derivatisierung ebenfalls zu realisieren. Insbesondere sollten die Derivate eine gute MS-Response aufweisen, um somit Verlusten durch unvollständige Umsetzung, mit denen bei allen Derivatisierungen immer gerechnet werden muss, entgegenzuwirken. Auch die Interpretation des Derivat-Leiterspektrums gegenüber dem underivatisierten Leiterspektrum sollte, falls möglich, vereinfacht sein. Tabelle 39 fasst die Eigenschaften der getesteten Derivatisierungsreagenzien zusammen.

Tabelle 39: Eigenschaften der untersuchten Derivatisierungsreagenzien

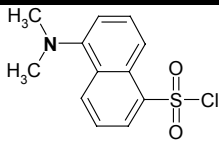
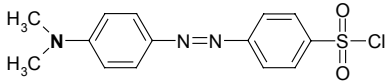
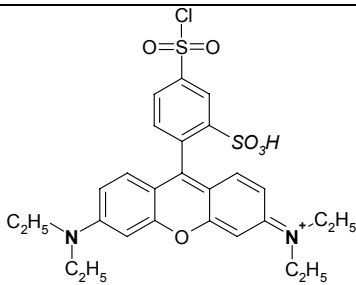
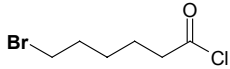
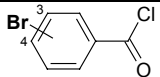
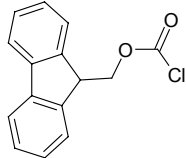
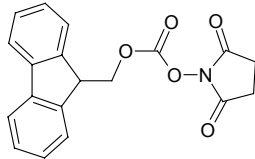
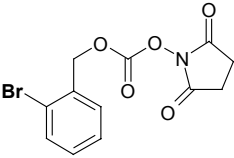
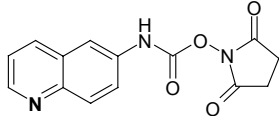
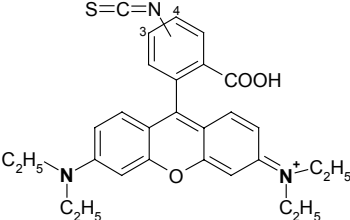
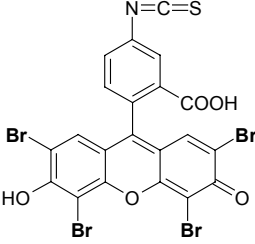
Name (verwendete Abkürzung)	Strukturformel	induzierter Massenshift [Da] ¹	spezielle Eigenschaften ²
Dansylchlorid 5-(Dimethylamino)- naphthalin-1-sulfonylchlorid <i>DNS-Cl</i>		233,0510	spezifische Absorption der Peptidderivate: $\lambda(\text{abs.}) = 340\text{nm}$
Dabsylchlorid 4-Dimethylaminoazobenzol- 4'-sulfonylchlorid <i>DABS-Cl</i>		287,0728	spezifische Absorption der Peptidderivate: $\lambda(\text{abs.}) = 436\text{nm}$
Sulforhodamin B Säurechlorid <i>SRB-Cl</i>		541,1467	spezifische Fluoreszenz der Peptidderivate: $\lambda(\text{abs.}) = 556\text{nm}$ $\lambda(\text{em.}) = 576\text{nm}$ „fixierte“, positive Ladung
6-Bromhexanoylchlorid <i>6-BH-Cl</i>		175,9837	charakteristisches Isotopenmuster der bromhaltigen Derivate
3-Brombenzoylchlorid und 4-Brombenzoylchlorid <i>3-BB-Cl und 4-BB-Cl</i>		181,9367	charakteristisches Isotopenmuster der bromhaltigen Derivate

Tabelle 39: Eigenschaften der untersuchten Derivatisierungsreagenzien (Fortsetzung)

Name (verwendete Abkürzung)	Strukturformel	induzierter Massenshift [Da] ¹	spezielle Eigenschaften ²
(9-Fluorenylmethyl)-chlorformiat <i>FMOC-Cl</i>		222,0681	spezifische Fluoreszenz der Peptidderivate: $\lambda(\text{abs.}) = 266\text{nm}$ $\lambda(\text{em.}) = 305\text{nm}$
(9-Fluorenylmethyl)-succinimidylcarbonat <i>FMOC-NHS</i>		222,0681	spezifische Fluoreszenz der Peptidderivate: $\lambda(\text{abs.}) = 266\text{nm}$ $\lambda(\text{em.}) = 305\text{nm}$
N-(2-Brombenzyloxy-carbonyl)-hydroxy-succinimid <i>BBOC-NHS</i>		211,9473	charakteristisches Isotopenmuster der bromhaltigen Derivate
6-Aminochinolyl-N-hydroxysuccinimid <i>ACQ™</i>		170,0480	spezifische Fluoreszenz der Peptidderivate: $\lambda(\text{abs.}) = 250\text{nm}$ $\lambda(\text{em.}) = 395\text{nm}$
Rhodamin B Isothiocyanat (Isomerengemisch) <i>RB-ITC</i>		500,2008	spezifische Fluoreszenz der Peptidderivate: $\lambda(\text{abs.}) = 556\text{nm}$ $\lambda(\text{em.}) = 576\text{nm}$ „fixierte“, positive Ladung
Eosin 5 Isothiocyanat <i>E5-ITC</i>		700,6778	spezifische Fluoreszenz der Peptidderivate: $\lambda(\text{abs.}) = 521\text{nm}$ $\lambda(\text{em.}) = 544\text{nm}$ charakteristisches Isotopenmuster der bromhaltigen Derivate

¹ monoisotopisch² $\lambda(\text{abs.})$: Absorptionswellenlänge; $\lambda(\text{em.})$: Emissionswellenlänge

Die Reagenzien lassen sich bezüglich ihrer reaktiven, funktionellen Gruppe wie folgt einteilen:

- Säurechloride (Sulfonsäure- und Carbonsäurechloride)
- Aktive Ester (N-Hydroxysuccinimide)
- Isothiocyanate

3.3.2.2 Selektivitäten und Produktspektren

Die folgenden beiden Tabellen geben im Überblick die Produktspektren einiger Testpeptide bei Umsetzungen mit den oben aufgeführten Derivatisierungsreagenzien wieder. Betrachtet wurden dabei, bis auf die Umsetzung mit SRB-Cl sowie mit den Isothiocyanaten RB-ITC und E5-ITC, die MALDI-Massenspektren der Rohprodukte. SRB-Cl, RB-ITC und E5-ITC erfordern eine RP-HPLC-Aufreinigung vor der MALDI-MS – die Rohprodukte ergeben keine auswertbaren Spektren. Die Intensitätswerte in den Tabellen sind jeweils auf das Derivat mit der höchsten Signalintensität im Spektrum normiert.

Eine Auswertung für die Tetrapeptide P4-492 (RLFG), P4-545 (RLWA), P4-487 (RLKA), P4-508 (RLYG) und P4-496 lässt Rückschlüsse auf die Selektivität der Umsetzung der einzelnen Reagenzien zu. Eine Derivatisierung des Tryptophan als Nebenreaktion wurde nicht beobachtet. Dagegen konnten unerwünschte Nebenreaktionen in unterschiedlichem Ausmaß mit den Aminosäuren His, Tyr und natürlich Lys beobachtet werden.

Tabelle 40: Produktspektren der Derivatisierungsreaktionen unterschiedlicher Peptide (Teil 1)

Peptid	Sequenz	relative Intensität					
		Edukt	Mono ¹	Di ²	Edukt	Mono ¹	Di ²
		DNS-Cl			DABS-Cl		
Bradykinin	RPPGFSPFR	-	100%	-	-	100%	-
P4-492	RLFG	-	100%	-	-	100%	-
P4-545	RLWA	-	-	-	-	100%	-
P4-487	RLKA	-	-	-	-	100%	31%
P4-508	RLYG	-	14%	100%	-	100%	58%
P4-496	RLHA	-	-	-	-	100%	15%
P10-1081	DFSALLSQIS	-	100%	-	-	-	-
P14-1624	ISSYQDAIEIELEN	-	37%	100%	-	-	-
		SRB-Cl			6-BH-Cl		
Bradykinin	RPPGFSPFR	-	100% ²	-	-	100%	-
P4-492	RLFG	-	100% ²	-	-	100%	-
P4-545	RLWA	-	100% ²	-	-	100%	-
P4-487	RLKA	-	100% ²	-	-		100%
P4-508	RLYG	-	100% ²	-	-	100%	41%
P4-496	RLHA	-	100% ²	-	-	48%	100%
P10-1081	DFSALLSQIS	-	-	-	-	100%	-
P14-1624	ISSYQDAIEIELEN	-	-	-	-	100%	-
		3-BB-Cl			4-BB-Cl		
Bradykinin	RPPGFSPFR	-	100%	18%	-	100%	8%
P4-492	RLFG	-	100%	-	-	100%	-
P4-545	RLWA	-	100%	-	-	100%	-
P4-487	RLKA	-	-	100%	-	-	100%
P4-508	RLYG	-	-	100%	-	-	100%
P4-496	RLHA	-	28%	100%	-	100%	82%
P10-1081	DFSALLSQIS	-	100%	-	-	100%	-
P14-1624	ISSYQDAIEIELEN	-	100%	20%	-	100%	-

Tabelle 43: Produktspektren der Derivatisierungsreaktionen unterschiedlicher Peptide (Teil 1; Fortsetzung)

Peptid	Sequenz	relative Intensität					
		Edukt	Mono ¹	Di ²	Edukt	Mono ¹	Di ²
		FMOC-Cl			FMOC-NHS		
Bradykinin	RPPGFSPFR	-	100%	-	9%	100%	-
P4-492	RLFG	-	100%	-	-	100%	-
P4-545	RLWA	-	100%	-	-	100%	-
P4-487	RLKA	-	-	100%	17%	100%	-
P4-508	RLYG	-	100%	81%	-	100%	-
P4-496	RLHA	-	100%	19%	-	100%	6%
P10-1081	DFSALLSQIS	-	100%	-	-	100%	-
P14-1624	ISSYQDAIEIELEN	-	-	-	82%	100%	-
		BBOC-NHS			ACQ		
Bradykinin	RPPGFSPFR	23%	100%	-	38%	100%	-
P4-492	RLFG	-	100%	-	-	100%	-
P4-545	RLWA	-	100%	-	34%	100%	-
P4-487	RLKA	-	7%	100%	-	100%	8%
P4-508	RLYG	-	100%	25%	-	100%	-
P4-496	RLHA	-	100%	57%	-	100%	-
P10-1081	DFSALLSQIS	-	100%	-	-	-	-
P14-1624	ISSYQDAIEIELEN	-	100%	-	100%	40%	-
		RB-ITC			E5-ITC		
Bradykinin	RPPGFSPFR	22%	100% ²	-	-	100% ²	-
P4-492	RLFG	-	100% ²	-	-	-	-
P4-545	RLWA	-	-	-	-	-	-
P4-487	RLKA	-	-	-	-	-	-
P4-508	RLYG	-	100% ²	-	-	-	-
P4-496	RLHA	-	100% ²	-	-	-	-
P10-1081	DFSALLSQIS	-	-	-	-	100% ²	-
P14-1624	ISSYQDAIEIELEN	-	-	-	-	-	-

¹ Mono = Monoderivat, Di = Diderivat² Auswertung erst nach HPLC-Aufreinigung – daher nur ein Produkt mit 100%

Ein Strich (-) bedeutet, dass das betreffende Edukt bzw. das Mono- oder Diderivat nicht nachweisbar war.

Insbesondere die Carbonsäurechloride 3-BB-Cl, 4-BB-Cl und 6-BH-Cl reagieren fast komplett mit allen drei oben genannten Aminosäuren. Das lysinhaltige Peptid RLKA wird, abgesehen von den Reaktionen mit den aktiven Estern FMOC-NHS und ACQ, von allen anderen Reagenzien vollständig am N-Terminus und an der ϵ -Aminogruppe des Lysin umgesetzt und liefert somit das Diderivat. Der Imidazolring des Histidins und die saure Hydroxylgruppe des Tyrosins sind nach den Resultaten der Umsetzung bezüglich ihrer Reaktivität hinter der freien Aminogruppen des N-Terminus oder des Lysins einzuordnen. Histidin und Tyrosin reagieren häufig nur unvollständig mit den Derivatisierungsreagenzien (z.B. DABS-Cl, FMOC-Cl oder BBOC-NHS). Für die Umsetzung der Peptide mit den beiden Isothiocyanaten RB-ITC und E5-ITC, sowie dem Säurechlorid von Sulforhodamin B

konnten weder in den Massenspektren der Rohprodukte, noch in den Fraktionen nach RP-HPLC-Aufreinigung Diderivate nachgewiesen werden. Im Fall der Umsetzung des Sulforhodamin B Säurechlorid mit den Peptiden RLKA und RLYG wurde jedoch nach HPLC Aufreinigung der Rohprodukte in jeweils zwei Fraktionen jedes Peptids ein Monoderivat nachgewiesen. Die Sequenzierung dieser beiden Monoderivate ergab, dass entweder der N-Terminus oder das Lysin beziehungsweise das Tyrosin derivatisiert wurden. Für die Tatsache, dass in diesen Fällen keine Diderivate beobachtet wurden, lassen sich zwei Begründungen in Erwägung zu ziehen, die beide ihre Ursache in der Hydrophobizität der Derivatisierungsreagenzien, respektive deren Derivate haben. Diese hohe Hydrophobizität zeigt sich deutlich im Retentionsverhalten bei der Aufreinigung mittels C18 RP-HPLC der Derivate. Entweder verhindert die abstoßende Wirkung der hydrophoben Reste die Bildung von Diderivaten weitgehend oder die starke Hydrophobizität entstandener Diderivate ist so hoch, dass eine Elution von der Säule mit den angewendeten Eluenten (bis 90% ACN beim Spülen) nicht mehr möglich ist. Die Resultate weiterer Derivatisierungen mit Peptiden komplexerer Funktionalität sind in Tabelle 41 zusammengestellt.

Tabelle 41: Produktspektren der Derivatisierungsreaktionen unterschiedlicher Peptide (Teil 2)

Peptid	Sequenz	relative Intensität					
		Edukt	Mono	Di	Tri	Tetra	Penta
DNS-P2-322	RF		100%				
DNS-P2-331	RR		100%	2%			
DNS-P3-487	RRR		100%	4%			
DNS-P8-997	KNYKESDI				17%	100%	
DNS-P9-1069	TKNYKQTSV				21%	100%	
DNS-P10-1191	NKDKNQESDI			11%	100%		
DNS-P10-1212	NKKKKDETV					12%	100%
DNS-P14-1639	REDFSGLLPEEFIS	24%	100%				
DNS-P14-1713	ESKFQQKLAFTTR			21%	100%		
DNS-P14-1720	QETFSDLWKLLPEN		100%				
DNS-Substance P	RPKPQQFFGLM-NH ₂			100%			
DNS-Neurotensin	(pyro) ELYENKPRRPYIL		1%	25%	100%		
DNS-Angiotensin 2	DRVYIHPF	27%	100%	94%	29%		
DNS-Angiotensin 1	DRVYIHPFHL		75%	100%	90%	49%	
ACQ-Angiotensin 1	DRVYIHPFHL	12%	100%				
DNS-ACTH 18-39	RPVKVYPNGAEDESAAFPLEF		100%	30%			
ACQ-ACTH 18-39	RPVKVYPNGAEDESAAFPLEF		100%	87%			
DABS-P4-548	RFIL		100%				
DABS-P5-593	TRGIF		100%				
DABS-P5-603	RESLV		100%				
DABS-P6-748	LSRFIL	61%	100%				

Diese Resultate bestätigen, zusammen mit denen aus Tabelle 40, dass Aminosäuren mit aliphatischen Hydroxylgruppen entweder nicht derivatisiert werden oder die entsprechenden Produkte der Umsetzung mit aliphatischen Hydroxylgruppen nicht stabil sind. Auch die Guanidiniumgruppe des Arginins verhält sich gegenüber den eingesetzten Reagenzien offensichtlich indifferent. Bei allen lysinhaltigen Peptiden mit freiem N-Terminus entspricht dagegen der höchste Derivatisierungsgrad der Zahl der enthaltenen Lysine plus eins. Zugleich weist dieser Derivatisierungsgrad in allen Fällen das intensivste Signal auf. Ob Tyrosin oder Histidin derivatisiert werden, wie z.B. im Falle des Angiotensin 1 oder P4-508 (RLYG), hängt vom Derivatisierungsreagenz ab. Die Umsetzung mit einem aktiven Ester erscheint hier grundsätzlich selektiver und führt in entsprechend geringerem Ausmaß zu Nebenreaktionen mit Tyrosin oder Histidin als mit den Säurechloriden.

3.3.2.3 Sequenzierung der Peptidderivate

Mit Ausnahme der Peptidderivate des Eosin 5 Isothiocyanat konnte von allen Umsetzungsprodukten mit den übrigen, in Tabelle 39 aufgeführten Derivatisierungsreagenzien mindestens ein Peptidderivat auch sequenziert werden. In der nachfolgenden Tabelle sind exemplarisch die Sequenzierungsergebnisse verschiedener Peptidderivate mit CPY dargestellt. Dabei handelt es sich um HPLC-gereinigte Dabsyl- und Sulforhodamin B-Derivate.

Tabelle 42: C-terminal sequenzierte Peptidderivate von Dabsylchlorid und Sulforhodamin B Säurechlorid (Der aus dem Leiterspektrum lesbare Teil der abgebauten Sequenz ist unterstrichen.)

Peptid	Derivatisierung mit DABS-Cl		Derivatisierung mit SRB-Cl	
	Derivatmasse ¹ [Da]	Sequenz	Derivatmasse ¹ [Da]	Sequenz
P4-492	779	(DABS) - RL <u>FG</u>	1033	(SRB) - RL <u>FG</u>
P4-508	795 / 1083	(DABS) - RLY <u>G</u>	1050	(SRB) - RLY <u>G</u>
P4-487	774 / 1061	(DABS) - RL <u>KA</u>	1028	(SRB) - RL <u>KA</u>
P4-496	784	(DABS) - RL <u>HA</u>	1038	(SRB) - RL <u>HA</u>
P4-545	832	(DABS) - RL <u>WA</u>	1086	(SRB) - RL <u>WA</u>
P4-548	836	(DABS) - RF <u>IL</u>	1089	(SRB) - RF <u>IL</u>
P5-593	881	(DABS) - TRG <u>IF</u>	1133	(SRB) - TRG <u>IF</u>
P5-603	891	(DABS) - RESL <u>V</u>	1144	(SRB) - RESL <u>V</u>
P6-748	1036	(DABS) - LSRF <u>IL</u>	1289	(SRB) - LSRF <u>IL</u>
Bradykinin	1348	(DABS) - RPPGFSP <u>FR</u>	1601	(SRB) - RPPGFSP <u>FR</u>

¹ Üblicherweise ist nur die Masse des sequenzierten Monoderivats angegeben, für die Umsetzung der Peptide P4-508 und P4-487 mit DABS-Cl wurde die Mischung aus Mono- und Diderivat sequenziert

Tetrapeptide („P4-Peptide“ nach Nomenklatorschema Punkt 2.1.3) liefern in der Regel bei einer Sequenzierung ohne vorherige Derivatisierung aufgrund ihrer geringen Masse (Interferenz mit Signalen der Matrixionen) keine verwertbaren Leiterspektren und somit auch keine Sequenzinformation. Diese „P4-Peptide“ lassen sich in Form geeigneter Derivate aber meist bis zum Dipeptid abbauen und erlauben auch ein Auslesen dieser Information aus dem Leiterspektrum. Ebenso kann bei den etwas größeren „P5- und P6-Peptiden“ der Abbau bis zu Di- oder Tripeptid verfolgt werden. Abbildung 46 zeigt zwei Beispiele für die Leitersequenzierung kleiner Peptide nach Derivatisierung Sulforhodamin B Säurechlorid. Bei einer mittleren Aminosäuremasse um 110 Da liegt die minimale Masse eines N-terminalen Leiterpeptidderivats mit einem der Reagenzien aus Tabelle 39 bei ca. 300 Da wenn die

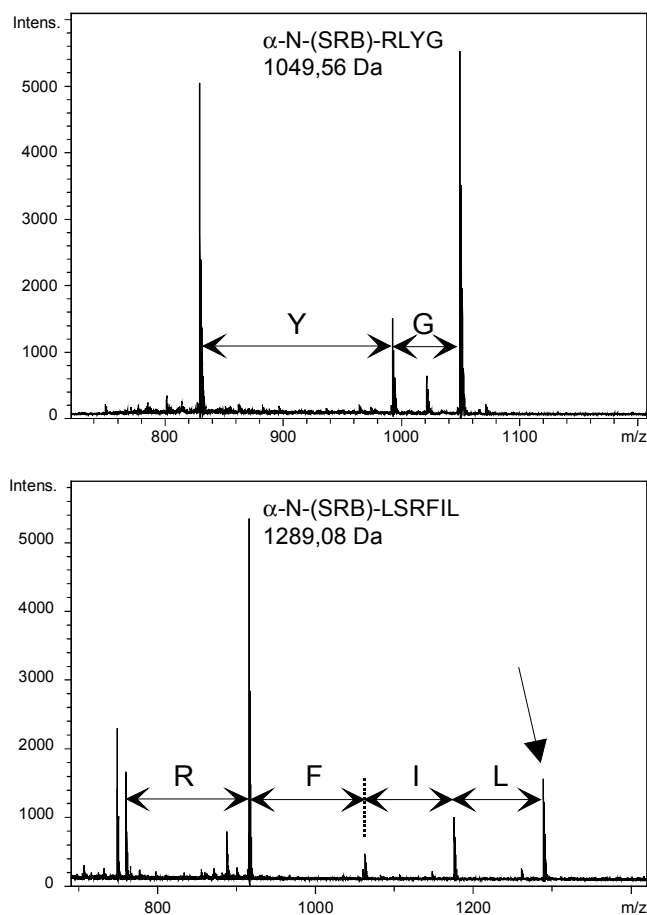


Abbildung 46: Beispiele für die Sequenzierung kurzer Peptide nach Derivatisierung;

oben: ca. 500fmol α -N-(SRB)-P4-408, sequenziert mit CPY-I;

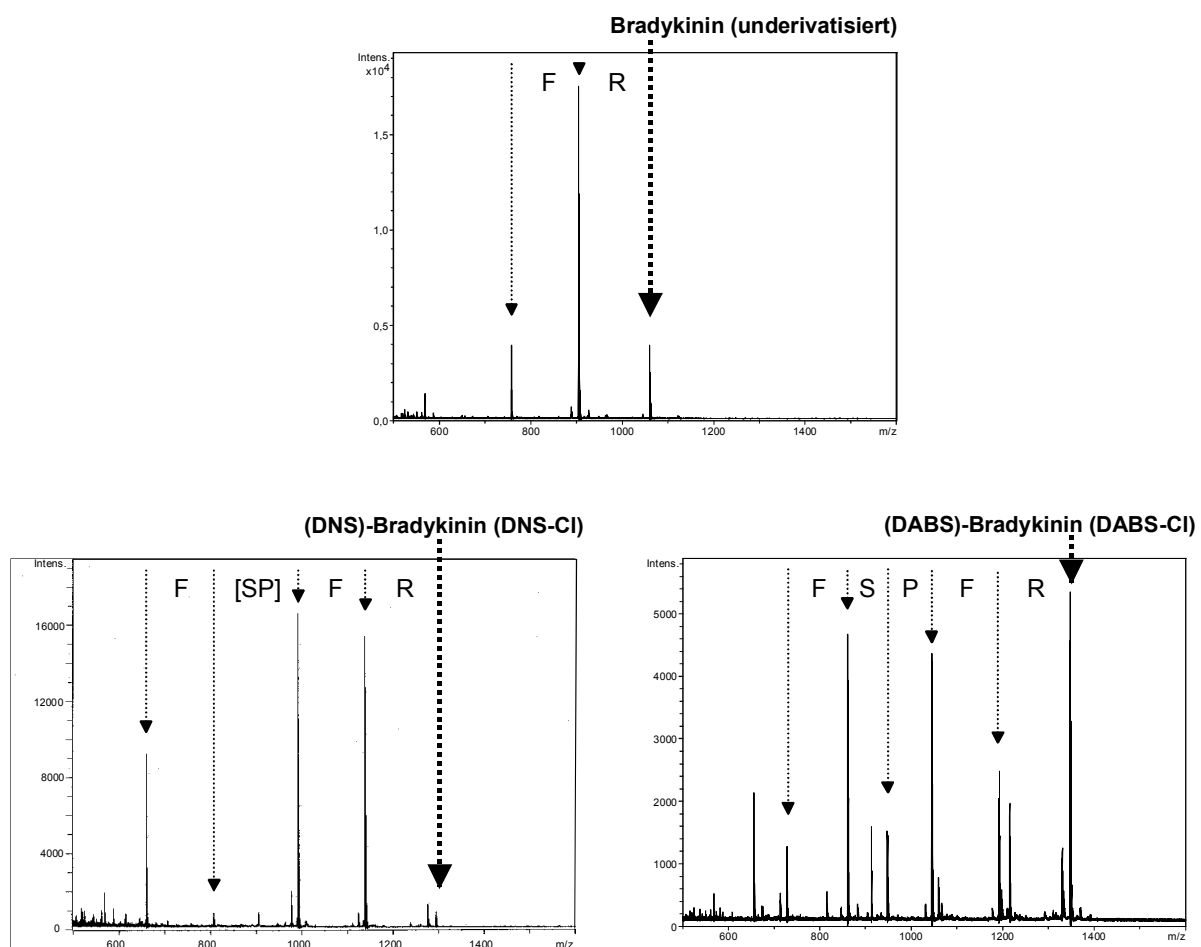
unten: zusammengesetztes Spektrum der Leitersequenzierungen von ca. 500fmol α -N-(SRB)-6-748 mit CPA-I und CPB

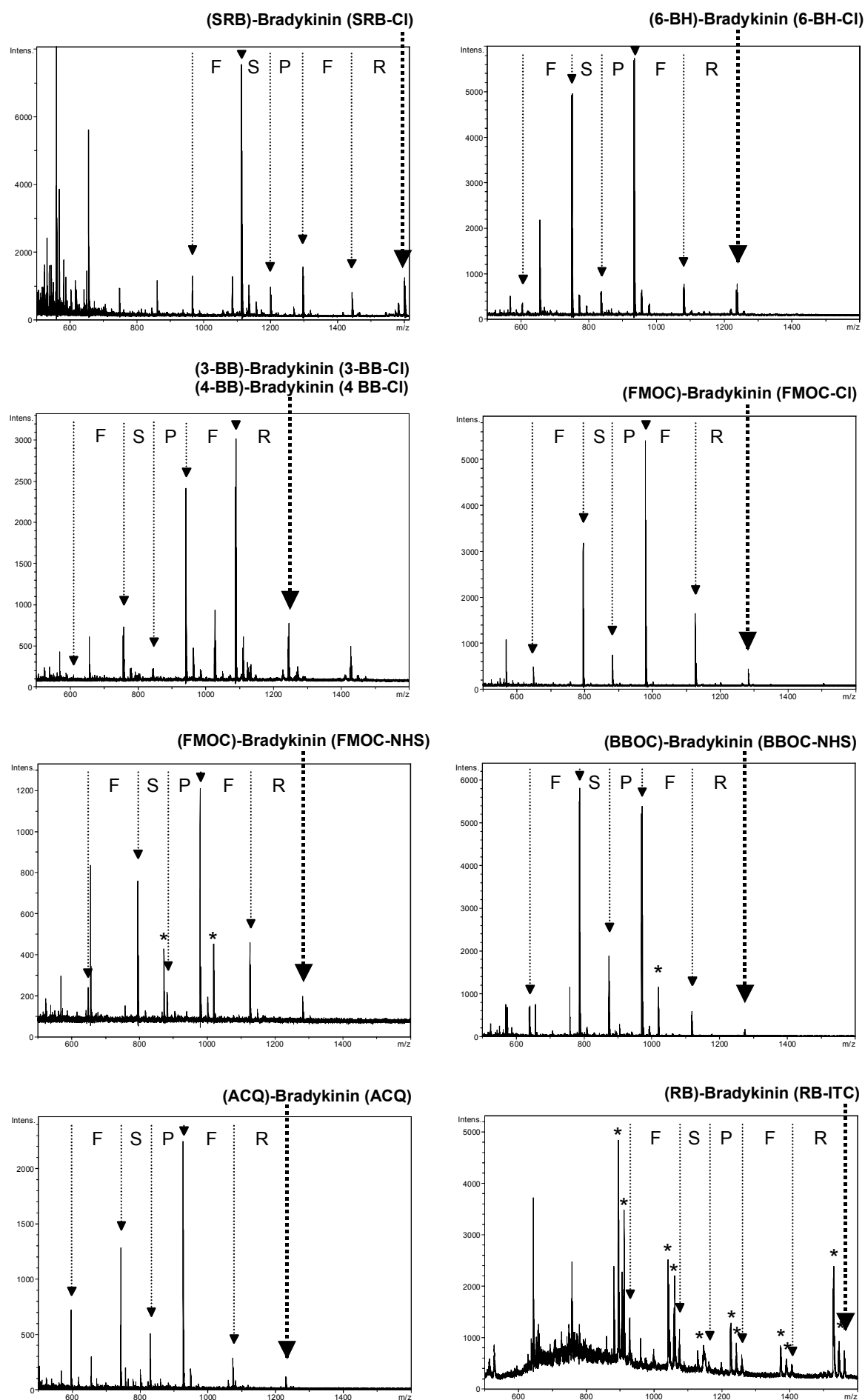
C-terminale Sequenzierung bis zur letzten Aminosäure erfolgt. Die Massen der häufig verwendeten Matrices CHCA und DHB liegen zwar mit $[M+H]^+$ von 190 Da und 155 Da deutlich darunter, allerdings zeigen für diese Matrices die Dimere mit 379 Da für CHCA und 307 Da eine sehr starke Signalintensität. Daher ist für einige der getesteten Reagenzien *a priori* eine Auswertung der Leiterspektren theoretisch nur bis zum Di- oder Tripeptid möglich. Bei SRB-Derivaten könnte man zumindest in einzelnen Fällen die Beobachtung eines kompletten Abbaus erwarten, da hier die Massenverschiebung weit über den Matrixbereich hinausgeht (z.B. in Abbildung 46 bei SRB-RLYG). Tatsächlich wird jedoch immer nur ein Abbau bis zum derivatisierten Dipeptid beobachtet. Da die getesteten

Carboxypeptidasen offensichtlich nicht zu einem Aminosäureabbau bei Dipeptiden befähigt sind, müsste hier ein weiterer Abbau über eine Dipeptidase (EC 3.4.13) erfolgen. Bei Dipeptidasen ist auch der Abbau N-terminal derivatisierter Dipeptide bekannt. Unter der Voraussetzung, dass der Abbau jedoch nur bis zum Dipeptid (mittlere Aminosäuremasse um 110 Da) erfolgt, ist selbst die durch ACQ eingeführte, sehr kleine Massenverschiebung von 170 Da in der Regel ausreichend.

Die nachfolgenden Spektren in Abbildung 47 zeigen Sequenzierungen von unterschiedlichen Bradykininderivaten. Alle Bradykininderivate erlauben im Gegensatz zum underivatisierten Bradykinin ein Auslesen der Sequenzinformation bis zum Tetrapeptid (beobachteter Abbau von FSPFR). Aus dem Leiterspektrum für das underivatisierte Peptid wurde dagegen lediglich die Information über den Abbau von zwei Aminosäuren erhalten (kurze Sequenz FR).

Abbildung 47: Vergleich des Leiterspektrums von underivatisiertem Bradykinin (RPPGFSPFR) mit Leiterspektren von Bradykininderivaten. Sequenzierung mit CPY-I, CPY-II und sb(CP-I); Alle gezeigten Spektren entsprechen einer Summe der Resultate (Spektrenüberlagerung); *: Derivat-Zersetzungsprodukt





3.3.2.4 Eigenschaften der Peptidderivate

Tabelle 43 fasst die Eigenschaften der Peptidderivate zusammen. Im Bezug auf die nachfolgende Sequenzierung ist die Stabilität in Lösung für nahezu alle Derivate unkritisch. Selbst nach über einem Monat bei +4°C Lagerungstemperatur kann für Derivate von Säurechloriden und N-Hydroxysuccinimiden noch keine Zersetzung festgestellt werden.

Tabelle 43: Eigenschaften der Peptidderivate

Reagenz	Nebenreaktion mit	MS-Response ¹	Stabilität Lösung	Stabilität MALDI-MS	Sequenzierung ³
DNS-Cl	Lys, Tyr, His	++	> 1 Monat	+	HPLC
DABS-Cl	Lys, Tyr, His	++	> 1 Monat	Fragment -132 Da	HPLC
SRB-Cl	Lys, Tyr	++	> 1 Monat	+	HPLC
6-BH-Cl	Lys, Tyr, His	+	> 1 Monat	+	Rohprodukt
3-BB-Cl	Lys, Tyr, His	+	> 1 Monat	+	Rohprodukt
4-BB-Cl	Lys, Tyr, His	+	> 1 Monat	+	Rohprodukt
FMOC-Cl	Lys, Tyr, His	+	> 1 Monat	+	Rohprodukt
FMOC-NHS	Lys	++	> 1 Monat	Fragment -107 Da	Rohprodukt
BBOC-NHS	Lys, Tyr	++	> 1 Monat	Fragment -97 Da	Rohprodukt
ACQ	Lys	++	> 1 Monat	+	Rohprodukt
RB-ITC	Lys	0	²	²	HPLC
E5-ITC	Lys	0	²	²	HPLC

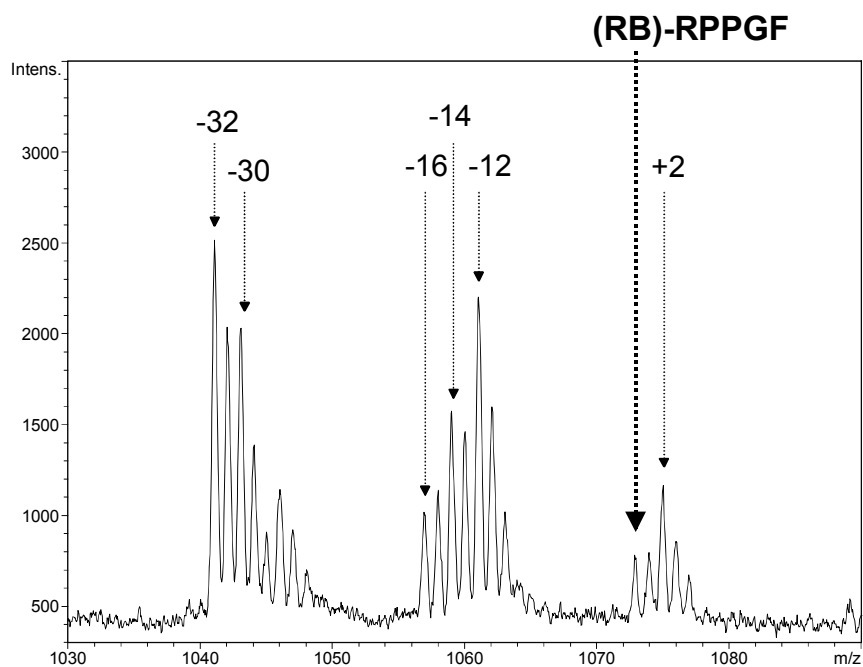
¹ Bezieht sich auf die MS-Response im Vergleich zum underivatisierten Peptid bei gleicher Menge für die MALDI-MS Messung; Dabei bedeuten: ++ wesentlich besser als / + besser als / 0 vergleichbar dem underivatisierten Peptid

² Ein einheitliches Produkt kann weder direkt nach der Derivatisierung aus dem Rohprodukt, noch aus der HPLC-Fraktion der Aufreinigung festgestellt werden.

³ HPLC: Aufreinigung vor der Sequenzierung erforderlich ; Rohprodukt: die Sequenzierung kann mit dem Rohprodukt erfolgen;

Die Umsetzungen von Peptiden mit Isothiocyanaten zeigen im Rohprodukt starke Nebensignale im Abstand von +2 Da bis –32 Da zum erwarteten Derivatisierungsprodukt (Abbildung 48). Ob eine Zersetzung bereits in Lösung oder erst während des MALDI-Prozesses eintritt kann aus den Resultaten nicht abgeleitet werden. Isothiocyanate haben sich in der Edman-Chemie als äußerst gute Reagenzien für die selektive Derivatisierung von Aminogruppen erwiesen und wären daher auch für die Peptidderivatisierung im Zusammenhang mit der Leitersequenzierung interessant. Leider zeigen auch HPLC-gereinigte Derivate ein Zersetzungsspektrum analog Abbildung 48. Die starke Zersetzung dürfte wohl auch dafür verantwortlich sein, dass die MS-Response der Isothiocyanate im Vergleich zu allen anderen Derivaten äußerst schlecht ausfällt.

Abbildung 48: MALDI-MS einer C-terminalen Leitersequenzierung von Rhodamin B - Bradykinin – Detailansicht des Spektrums des Leiterpeptids (RB)-RPPGF und seiner möglicher Zersetzungsprodukte.



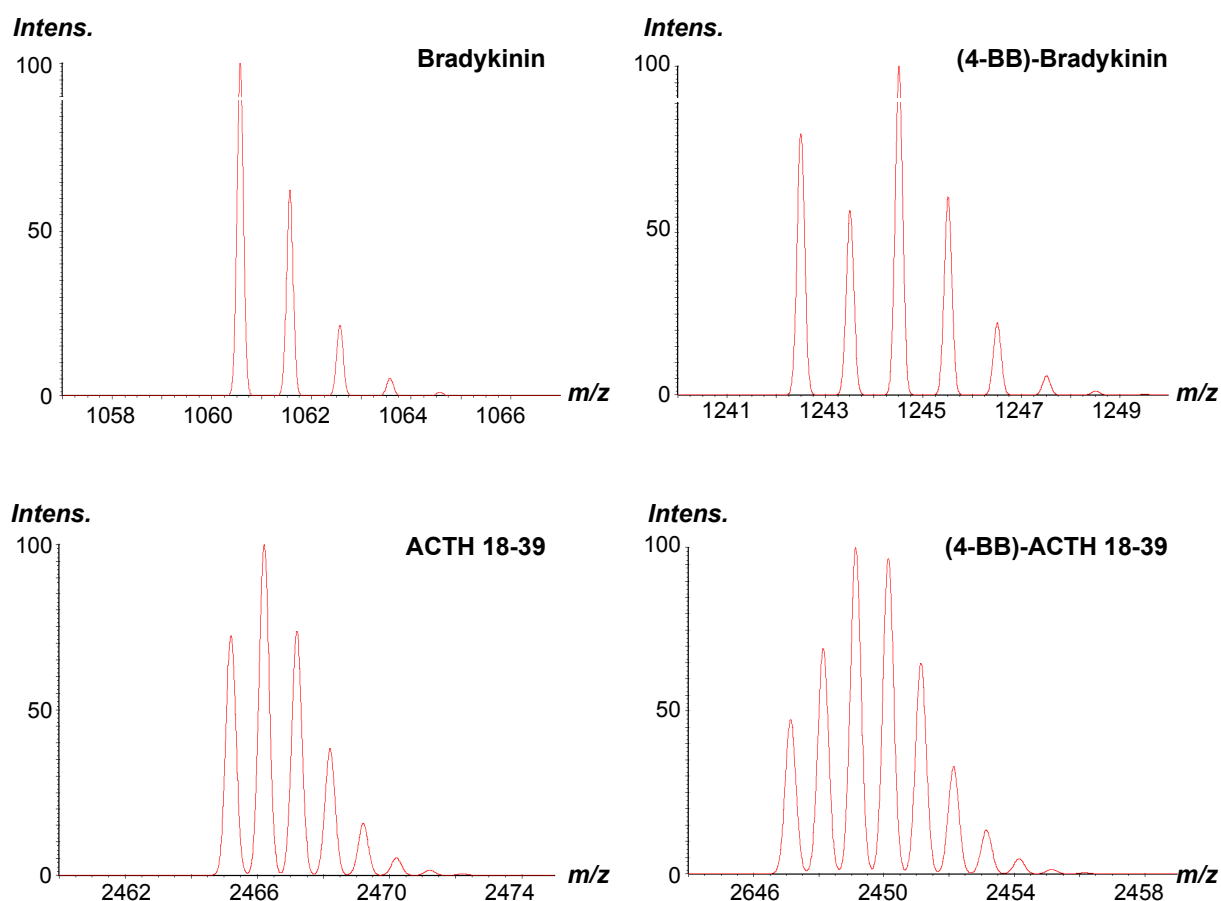
Derivate der Säurechloride und N-Hydroxysuccinimide erscheinen demgegenüber nicht nur in Lösung, sondern auch im MALDI-Prozess als relativ stabil. Tabelle 43 zeigt aber, dass auch für Derivate von DABS-Cl, sowie FMOC-NHS und BBOC-NHS das Auftreten einer charakteristischen Massendifferenz im MALDI-MS beobachtet werden kann. Für jeden Peak eines derivatisierte (Leiter-)Peptids im Massenspektrum existiert also ein Peak mit einer definierten, gleichbleibenden Massendifferenz. Allerdings liegen die Signalintensitäten der Säurechlorid- und N-Hydroxysuccinimidderivate in allen Fällen, selbst bei Messung der Rohprodukte, über den Signalintensitäten der Ausgangsprodukte. Besonders deutlich fallen die guten Signalintensitäten der Dansyl-, Dabsyl- und Sulforhodamin B-Derivate auf.

Tabelle 43 zeigt auch, dass ein Großteil der Derivate bereits eine Sequenzierung der entsprechenden Rohprodukte erlaubt. Dabei kann das Rohprodukt direkt aus der Reaktionsmischung heraus auf das Target gegeben und anschließend durch Auftragung der Peptidase sequenziert werden. Die im Falle von Dansyl-, Dabsyl- und SRB-Derivaten notwendige HPLC-Aufreinigung wird durch Detektion bei den charakteristischen Absorptionswellenlängen von 340nm für DNS-Derivate, 436nm für DABS-Derivate oder 556nm bei SRB-Derivaten stark vereinfacht. Auch für andere Peptidderivate ist zum Teil eine einfache und empfindliche Detektion nach HPLC-Reinigung über spezifische Absorptions- oder Fluoreszenzmaxima (Peptidderivate der Umsetzung mit ACQ, FMOC-NHS, Sulforhodamin B Säurechlorid sowie Rhodamin B Isothiocyanat und Eosin 5 Isothiocyanat) möglich.

3.3.3 Bromhaltige Peptidderivate

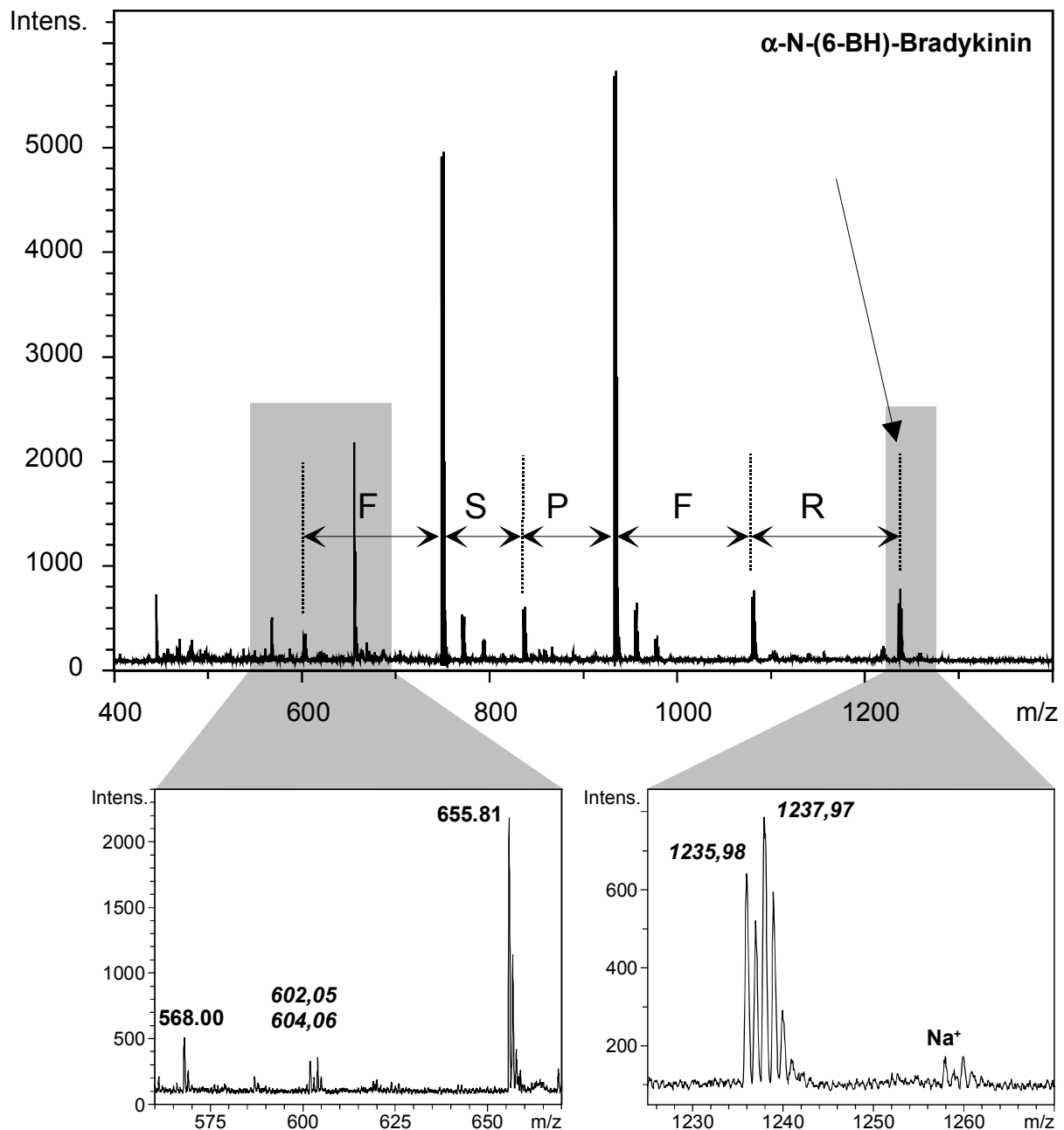
Durch die Derivatisierungsreagenzien 3-BB-Cl, 4-BB-Cl, 6-BH-Cl, BBOC-NHS, sowie E5-ITC wird Brom in das Peptidderivat eingeführt. Der Vorteil solcher bromhaltigen α -N-Derivate liegt in der Tatsache, dass auch die entsprechenden Leiterpeptide des Derivats durch ihr charakteristisches Isotopenmuster sehr leicht im Massenspektrum erkannt werden können. Eine Unterscheidung von anderen, nicht zur Leiter gehörenden Signalen (z.B. von Salzen) ist somit besser möglich. Abbildung 49 zeigt das typische Isotopenmuster eines Peptids mit einer Masse um 1000 Da (Bradykinin) und eines Peptids mit einer Masse zwischen 2000 Da und 3000 Da (ACTH Fragment 18-39), sowie die Isotopenmuster der monoderivatisierten α -N-4-Brombenzoyl-Derivate dieser Peptide.

Abbildung 49: Simulierte Isotopenmuster (Auflösung: 5000) der Peptide Bradykinin (oben) und ACTH (unten). Links underivatisiert, rechts derivatisiert mit 4-Brombenzoylchlorid (4-BB-Cl).



In der Praxis können solche Isotopenmuster das Auslesen der Leiter aus dem Spektrum stark vereinfachen, wie auch Abbildung 50 am Beispiel einer C-terminalen Leiter des α -N-(6-BH)-Bradykinins (Rohprodukt) zeigt.

Abbildung 50: Leiterspektrum von 500fmol α -N-(6-BH)-Bradykinin nach Sequenzierung mit CPY₁ (5min, 30°C)



Selbst das sehr intensitätsschwache Leiterpeptidsignal bei 602/604 Da kann als solches aufgrund des Isotopenmusters eindeutig von den wesentlich stärkeren Signalen bei 568 Da und 655 Da unterschieden und somit der Leiter zugeordnet werden. Analoge Resultate wurde für die Rohprodukte von α -N-(3-BB)-Bradykinin, α -N-(4-BB)-Bradykinin und α -N-(BBOC)-Bradykinin erhalten.

Die Tatsache, dass sich bromhaltige Derivate einer Peptidleiter im MALDI-Leiterspektrum leicht von anderen, nicht zur Leiter gehörigen Signalen unterscheiden lassen, spiegelt sich auch in der Empfindlichkeit wider.

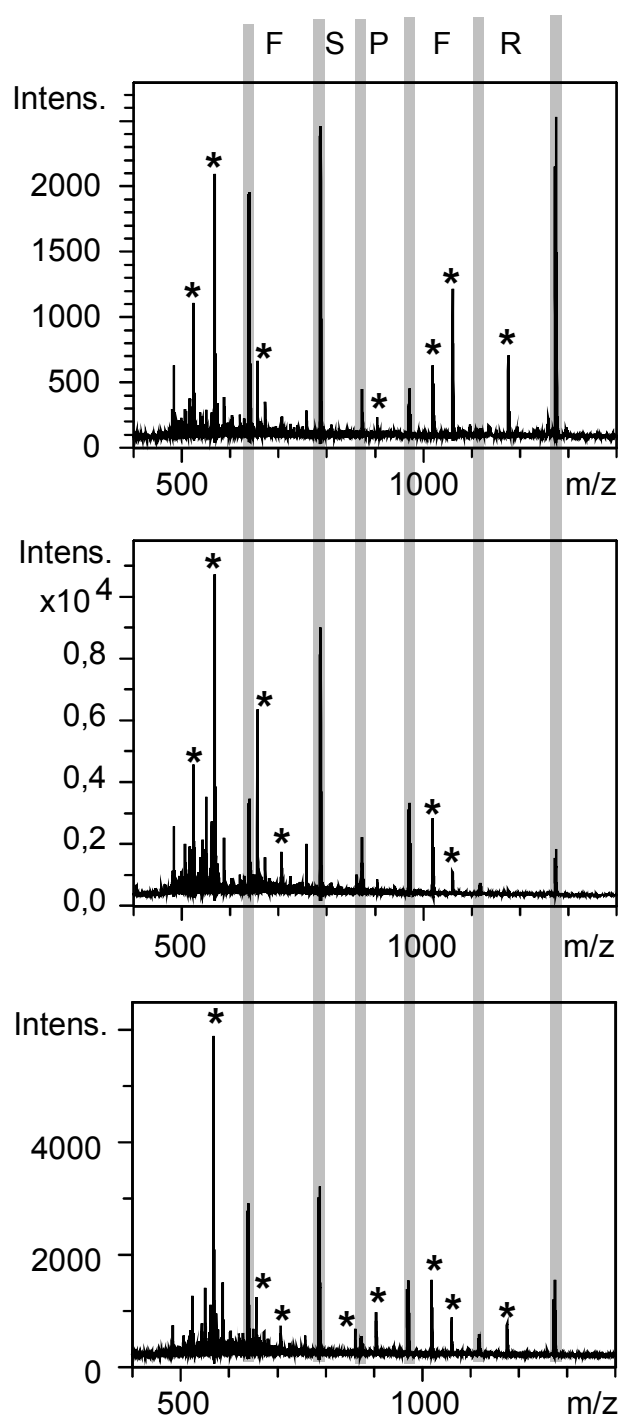


Abbildung 51 zeigt Leiterspektren von α -N-(BBOC)-Bradykininderivaten. Derivatisiert wurden 500 fmol, anschließend wurden Verdünnungen des Rohprodukts (250 fmol, 50 fmol und 15 fmol) auf das Target aufgetragen und mit CPY sequenziert. In allen Fällen erlaubt das charakteristische Isotopenmuster der Leiterpeptide trotz vieler Störsignale (*) – ähnlich wie in Abbildung 50 – ein eindeutiges Auslesen der Peptidleiter, selbst bei 15 fmol. Einige intensive Nebensignale kommen auch von Zersetzungsprodukten, die zum zugehörigen Leiterpeptid einen Abstand von -97 Da (siehe Tabelle 43) haben. Diese Satellitensignale sind zwar ebenfalls bromhaltig, jedoch immer äquidistant zur eigentlichen Peptidleiter und stören somit die Auswertung nicht wesentlich.

Abbildung 51: Leiterspektren von α -N-(BBOC)-Bradykininderivaten (Rohprodukte) generiert mit CPY₁ bei 30°C, 5 min

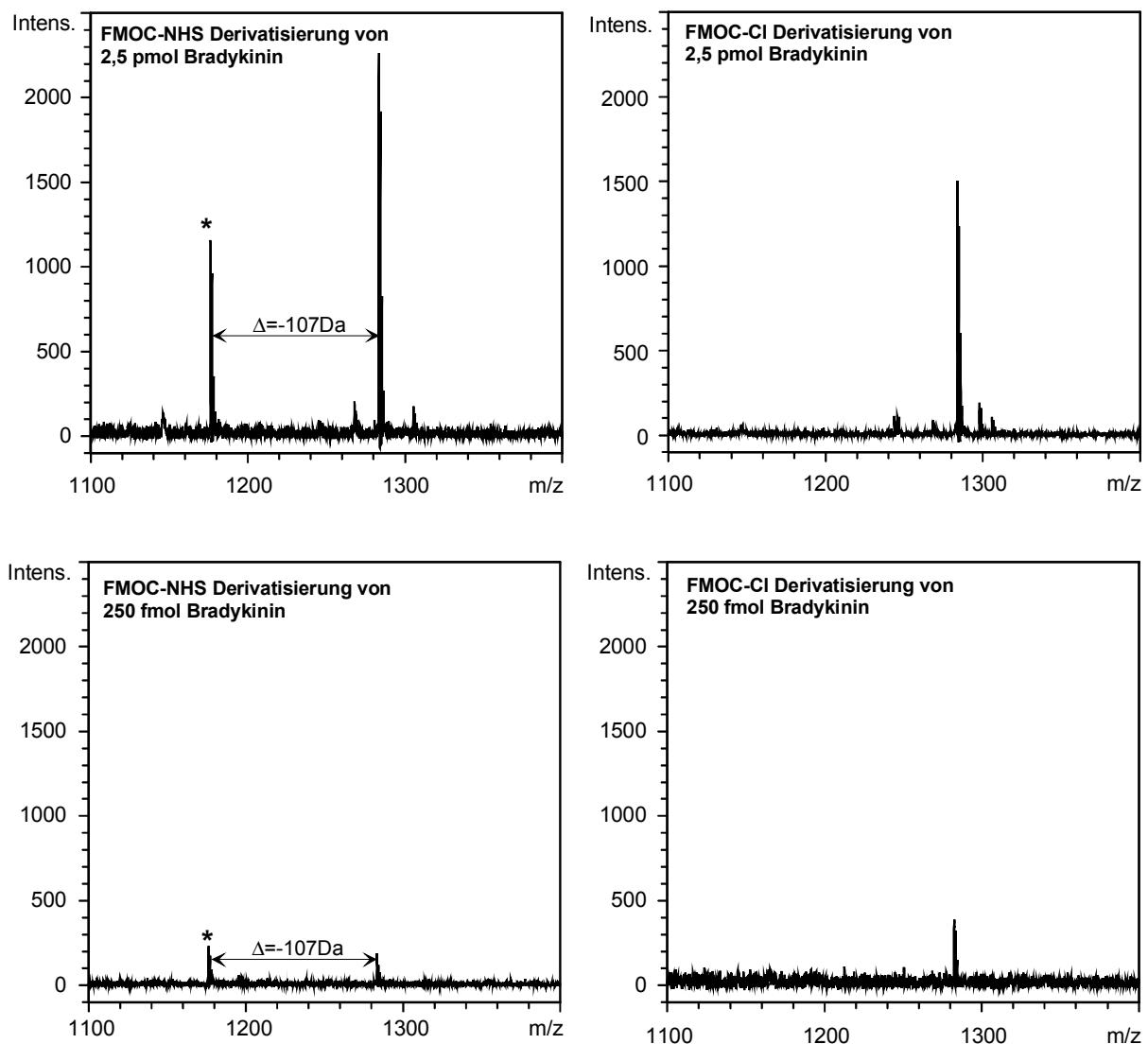
Nur die grau unterlegten Peaks gehören zur Leiter. Sequenzierung von 250 fmol (oben), 50 fmol (Mitte) und 15 fmol (unten)

3.3.4 Erforderliche Peptidmengen für die Derivatisierungen

Anhand von Umsetzungen des Bradykinins mit FMOC-NHS und FMOC-Cl sollten notwendige Mindestmengen an Peptid für eine Derivatisierung ermittelt werden, die anschließend auch noch eine erfolgreiche Detektion der Derivatisierungsprodukte erlauben. Die Resultate in Abbildung 52 zeigen, dass diese Peptidmengen im unteren Pikomol-Bereich beziehungsweise oberen Femtomol-Bereich liegen.

Alle Spektren in Abbildung 52 stammen von Rohprodukten einer Derivatisierung in Lösung. Ein Aliquot des jeweiligen Ansatzes wurde direkt auf das MALDI-Target aufgetragen. Die Spektren zeigen, dass die notwendige Peptid-Ausgangsmenge für eine Derivatisierung mit anschließend erfolgreicher Detektion im oberen Femtomol-Bereich (250fmol) liegt. Der entnommene Aliquot (1,0 - 0,5 μ l) eines solchen Reaktionsansatzes für die MALDI-Präparation enthält theoretisch bis zu 25 - 2,5fmol Peptidderivat (bei 100% Reaktionsumsatz).

Abbildung 52: Peptidmindestmengen für Derivatisierungsreaktionen; Bei den Derivatisierungen von 2,5pmol Bradykinin (oben) wurden 25fmol (FMOC-NHS), bzw. 10fmol (FMOC-Cl) des Derivats auf das MALDI-Target aufgetragen. Bei den Derivatisierungen von 250fmol Bradykinin (unten) wurden 5fmol (FMOC-NHS), bzw. 2,5fmol (FMOC-Cl) des Derivats auf das MALDI-Target aufgetragen.



3.4 Post-translationale Modifikationen

Aus dem bisherigen Resultaten geht hervor, dass artifiziell modifizierte Aminosäuren, wie z.B. acetyliertes Lysin durch Carboxypeptidasen oder auch pyridinethyliertes Cystein durch Carboxy- oder Aminopeptidasen prinzipiell abgebaut werden können. Eine Modifikation in der Seitenkette einer Aminosäure stellt also kein grundsätzliches Hindernis für die Abbaubarkeit dieser Aminosäure dar – auch wenn die eingeführte Gruppe voluminöser ist, wie etwa der Phenylrest des Vinylpyridins. Eine enzymatische Leitersequenzierung stellt somit eine gangbare Möglichkeit zur Identifizierung post-translationaler Modifikationen dar, sofern die modifizierte Aminosäure eine eindeutige Masse besitzt.

Das bereits untersuchte ϵ -Acetyllysin (Punkt 3.3.1) stellt nicht nur eine künstliche Modifikation dar, sondern kommt in einigen Proteinen natürlich vor. Acetylierungen wie in ϵ -Acetyllysin oder auch Acetylserin stellen neben Methylierungen (z.B. ϵ -N,N,N-Trimethyllysin, 3-Methylhistidin, N,N,N-Trimethylalanin), Hydroxylierungen (4-Hydroxyprolin, 5-Hydroxylysin) oder C-terminalen Amidierungen wichtige Formen natürlich vorkommender, modifizierter Aminosäuren dar. Besonders interessant aus biologischer Sicht sind aber vor allem Phosphorylierungen und Glykosylierungen einzelner Aminosäuren.

Experimentell konnte aus dieser Vielfalt möglicher Modifikationen nur ein kleiner Teil eingehender untersucht werden. Genauere Sequenzierungsstudien erfolgten mit der wichtigen Gruppe der Phosphopeptide. Hier standen mit Phosphoserin- und Phosphotyrosinpeptiden eine Reihe definierter Ausgangsverbindungen zur Verfügung. Für jede Sequenzierung der modifizierten Peptide wurden die nachfolgend aufgeführten Einzelpeptidasen und Peptidasekombinationen getestet.

Tabelle 44: Angewendete Einzelpeptidasen und Peptidasekombinationen bei post-translational modifizierten Peptiden (Phosphopeptide)

Aminopeptidasen	Carboxypeptidasen
APM: APM ₁	CPY: CPY ₁ / CPY ₂
AAP: AAP ₁	CPP: CPP ₁ / CPP ₂
API: API ₁	CPW: CPW ₁ / CPW ₂
sb(AP): AAP + APM / AAP + API	CPA: CPA ₁
	sb(CP-I): CPY+CPP / CPP+CPY
	pb(CP): CPY+CPP

Tabelle 46 und Tabelle 45 geben zunächst die Sequenzierungsergebnisse für die Phosphopeptide, getrennt nach C- und N-terminaler Sequenzierung wieder. Der abgebaute Teil der Sequenz ist, wie in den vorangegangenen Kapiteln, unterstrichen. Diejenigen Peptidasen und Peptidasekombinationen, die bei den einzelnen Sequenzierungen zum Erfolg geführt haben stehen jeweils unter der Sequenz grau unterlegt. Sofern ein zum phosphorylierten Peptid entsprechendes, nicht-phosphoryliertes Peptid zur Verfügung stand, wurde dieses ebenfalls untersucht. Damit sollte geprüft werden, ob Probleme bei der Sequenzierung von phosphorylierten Peptiden direkt mit der Phosphorylierungsstelle zusammenhängen oder ob diese Probleme andere Ursachen haben.

Tabelle 46 zeigt, dass durch N-terminale Sequenzierung sowohl Phosphoserin-, als auch Phosphotyrosinpeptide erfolgreich sequenziert werden können. APM als einzeln angewendete Peptidase oder in Kombination führt dabei mit allen Peptiden zu einem Abbau der phosphorylierten Aminosäure. AAP und API als Einzelpeptidasen führen in ihren Anwendungen nur vereinzelt zum Erfolg.

Tabelle 45: C-terminale Sequenzierungsergebnisse von Phosphopeptiden

Bezeichnung	Sequenzresultat phosphoryliertes Peptid	Sequenzresultat nicht-phosphoryliertes Peptid
Cys-Heat Shock Protein 25kD, Fragment 81-93	CLNRQL S (phos) <u>SGVSE</u> [IR] CPY: CPY ₂ sb(CP-I): CPY+CPP	
phos-P15-1474 nicht phosphoryliert: Aldolase : Fragment 343-357 aus Spaltung mit Chymotrypsin	TPSGQAGAAASE S (phos) <u>LF</u> CPA ₁	TPSGQAGAA <u>[AS]</u> <u>ESLF</u> sb(CP-I): CPY+CPP pb(CP): CPY+CPP
phos-P10-1277 nicht phosphoryliert: P10-1197 ¹ nicht phosphoryliertes Analogon Alkoholdehydrogenase: Fragment 12-20 aus Spaltung mit Chymotrypsin	YESHGKLE Y (phos) <u>A</u> CPW ₂	YESHGKLE Y ¹ CPP ₁ Alkoholdehydrogenase Fragment 12-20 aus Spaltung mit Chymotrypsin <u>YESHGKLE</u> sb(CP-I): CPY+CPP
phos-P12-1440 nicht phosphoryliert: P12-1360-NH ₂	PTSFGY (phos) <u>DKPHVL-NH₂</u> CPY: CPY ₁ / CPY ₂ CPP: CPP ₁ / CPP ₂ sb(CP-I): CPY+CPP / CPP+CPY	PTSFGYDKPHVL-NH ₂ CPY: CPY ₁ / CPY ₂ CPP: CPP ₁ / CPP ₂ sb(CP-I): CPY+CPP / CPP+CPY
phos-P12-1445 nicht phosphoryliert: P12-1365-NH ₂	VVHSGY (phos) <u>RHQVPS-NH₂</u> CPY ₂ CPP ₂ sb(CP-I): CPY+CPP / CPP+CPY	VVHSGYRHQVPS-NH ₂ CPY ₂ CPP ₂ sb(CP-I): CPY+CPP / CPP+CPY

¹ erhalten über *on-target* Dephosphorylierung mittels alkalischer Phosphatase

Tabelle 46: N-terminale Sequenzierungsergebnisse von Phosphopeptiden

Bezeichnung	Sequenzresultat phosphoryliertes Peptid	Sequenzresultat nicht-phosphoryliertes Peptid
Cys-Heat Shock Protein 25kD, Fragment 81-93	CLNRQLS (phos) SGVSEIR APM ₁ sb(AP): AAP+APM	
phos-P13-1550 nicht phosphoryliert: Aldolase :Fragment 1-13 aus Spaltung mit Endoproteinase LysC	PHS (phos) HPALTPEQKK sb(AP) : AAP+APM	PHSHPALTPEQKK AAP ₁ API ₁ sb(AP) : AAP+APM
phos-P13-1560	LGEY (phos) GFQNALIVR APM ₁ sb(AP): AAP+APM / AAP+API	LGEYGFQNALIVR APM ₁ sb(AP): APM+AP
β-Casein : Fragment 48-63 aus Spaltung mit Endoproteinase LysC	FQS (phos) EEQQQTEDELQDK APM ₁	

Anmerkung: Die Phosphopeptide phos-P12-1440-NH₂ und phos-P12-1445-NH₂, sowie ihre nicht-phosphorylierten Analoga P12-1360-NH₂ und P12-1365-NH₂ wurden nicht N-terminal untersucht

Abbildung 53 und Abbildung 54 zeigen Beispiele für N-terminale Leiterpeptidspektren eines Tyrosin- und eines Serinphosphopeptids. Die Beobachtung, dass der Peak vor der phosphorylierten Aminosäure am intensivsten ist, wurde allgemein bei allen sequenzierten N-terminal sequenzierten Phosphopeptiden gemacht. Dies deutet auf ein kinetisch ungünstiges Abspaltungsverhalten der phosphorylierten Aminosäure hin. Die mit Stern (*) gekennzeichneten Satellitenpeaks sind charakteristische, metastabile Fragmente, die im MALDI-MS durch Abspaltung von Phosphat entstehen. Diese metastabilen Fragmente können bei der Auswertung der MALDI-MS Leiterspektren besonders hilfreich sein. Die genaue, apparente Massendifferenz Δm der beobachteten Fragmentationen zum jeweiligen Mutterion im Spektrum ist abhängig von den gewählten Spannungsparametern. Mit den verwendeten Parametersätzen für die Messung der Leiterpeptide liegt Δm im Bereich zwischen 87 - 88 Da. In Abbildung 55 beträgt Δm 87,7 Da und ist hier indikativ für die metastabile Abspaltung des Phosphats (Spannungsparameter: $U_{IS1}=20,0$ kV, $U_{IS2}=15,4$ kV und $U_{ref}=21,5$ kV).

Abbildung 53: N-terminal sequenziertes Tyrosin-Phosphopeptid phos-P13-1560 (APM+AP)

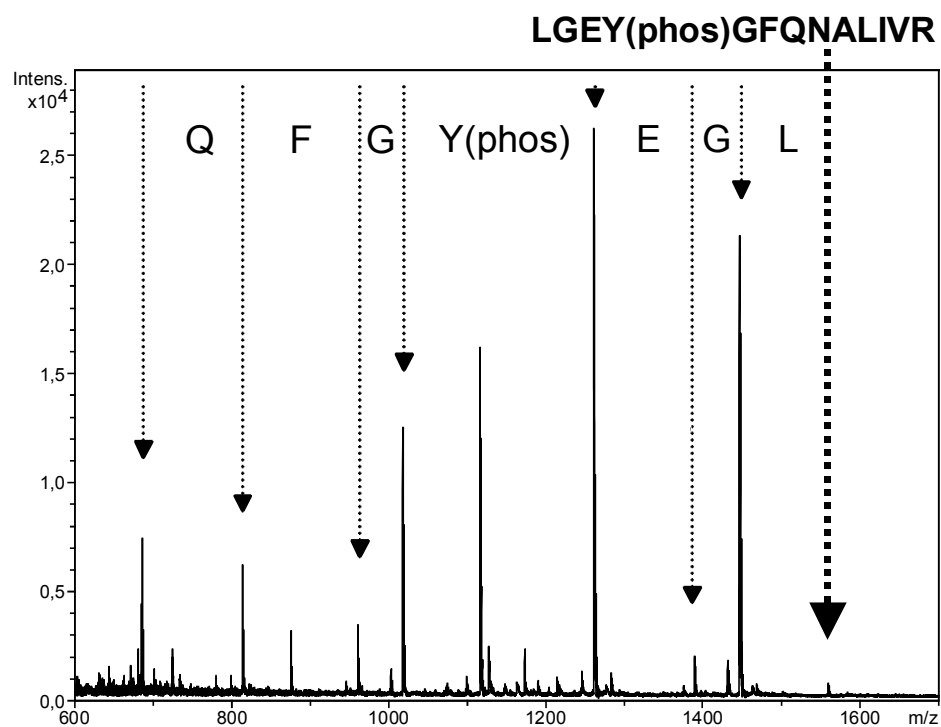


Abbildung 54: N-terminal sequenziertes Serin-Phosphopeptid phos-P13-1550 (AP+APM)

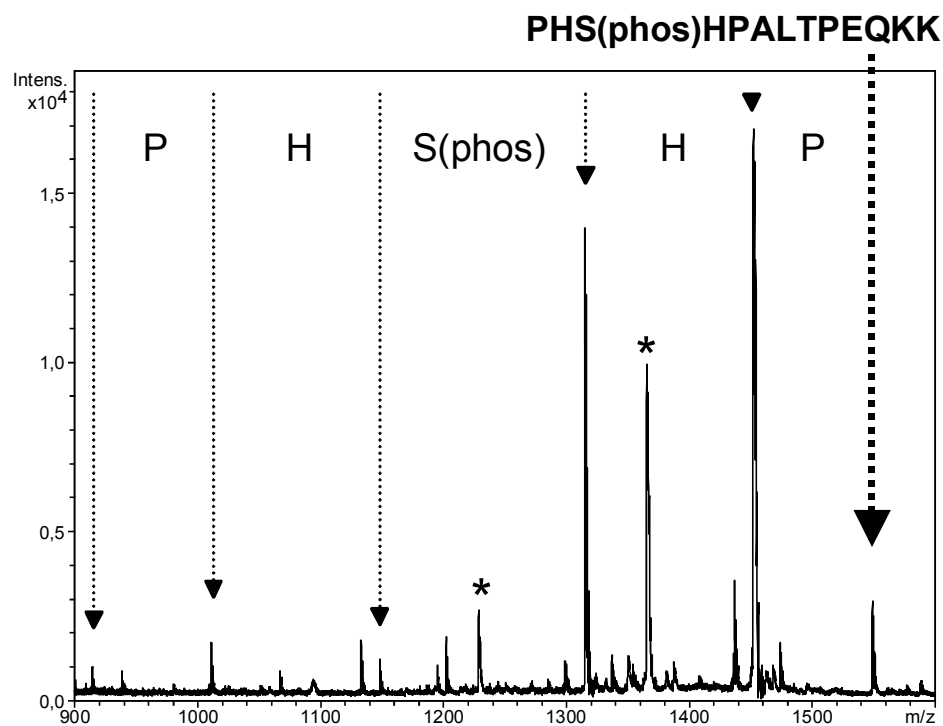
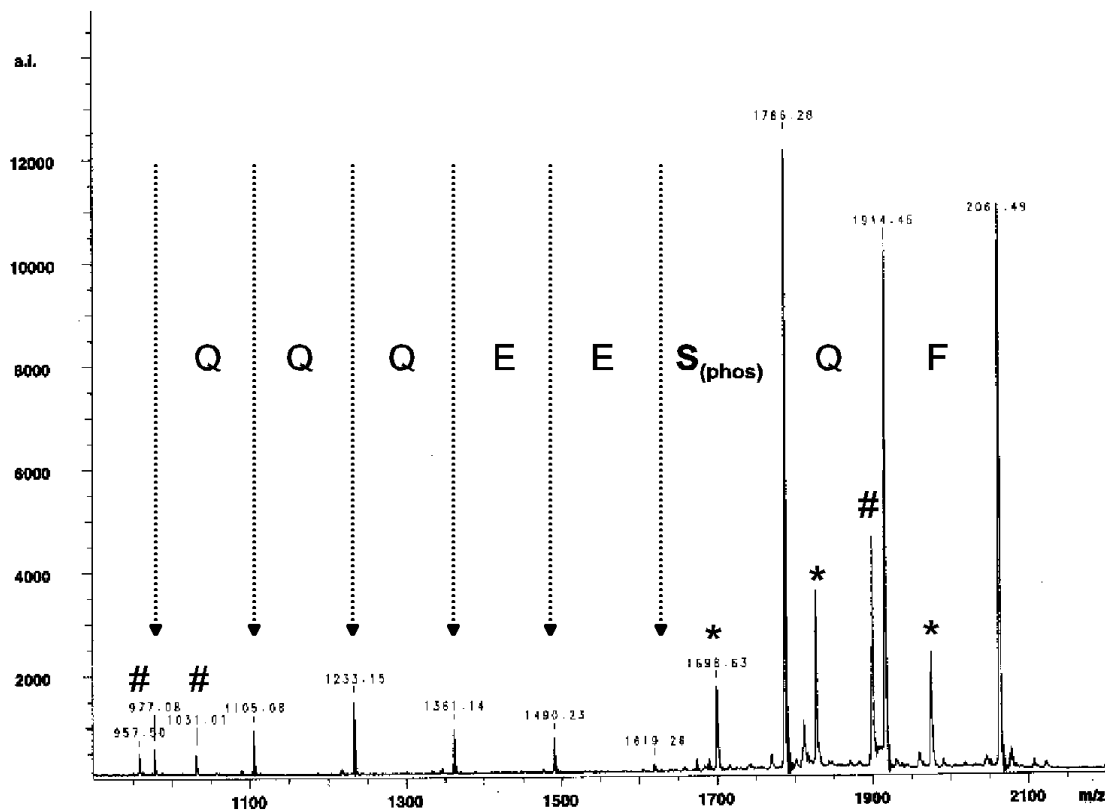


Abbildung 55: N-terminales Leiterspektrum von β -Casein Fragment 48-63 nach Spaltung mit APM₁ (Sequenz: FQS(phos)EEQQTEDELQDK)



Größere Probleme ergaben sich bei der C-terminalen Sequenzierung der Phosphopeptide. Hier konnte nur aus dem Peptid phos-P10-1277 das Phosphotyrosin nachweisbar abgebaut werden (siehe Abbildung 56). Phosphoserin konnte in keinem der untersuchten Fälle abgebaut werden. Gerade die erhaltene Teilsequenz für das Aldolase Fragment 343-357 (aus Spaltung mit Chymotrypsin) legt die Vermutung nahe, dass die phosphorylierte Aminosäure die Problemstelle des Abbaus darstellt, da das nicht-phosphorylierte Serin gut abgespalten wird. Aber auch der Abbau des Phosphotyrosins erscheint nach dem in Abbildung 56 gezeigten Spektrum kinetisch nicht günstig zu sein. Mit den Carboxypeptidasen CPY und CPP konnte in keinem der untersuchten Fälle ein Abbau einer Phospho-Aminosäure beobachtet werden. Der Abbau des Phosphotyrosins gelang mittels CPW.

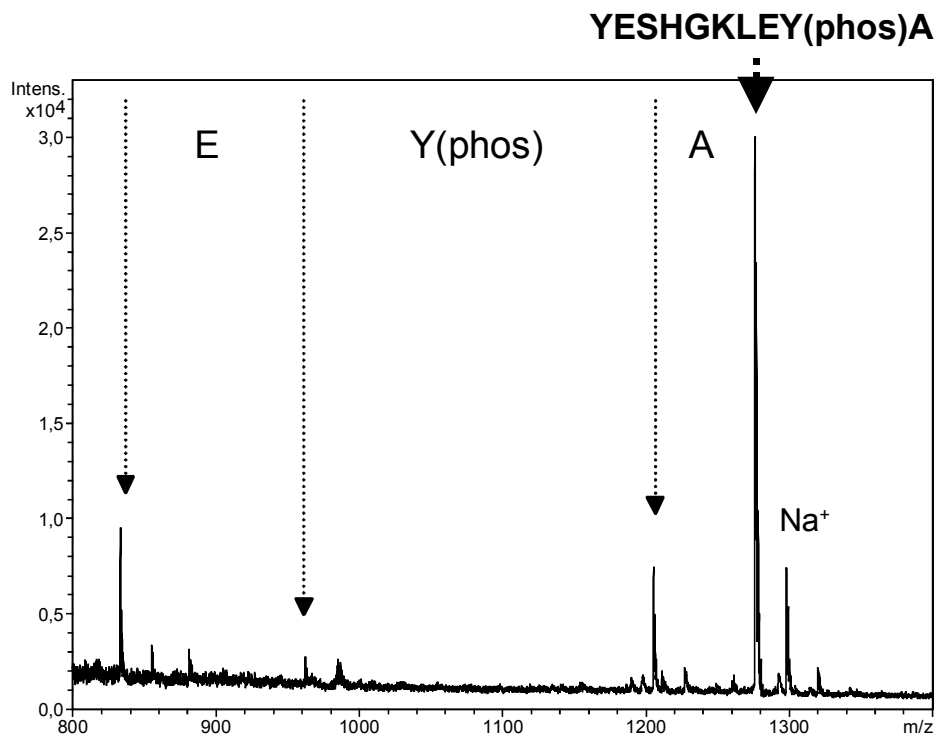
Abbildung 56: C-terminal sequenziertes Phosphopeptid phos-P10-1277 sequenziert mit CPW₂

Tabelle 47 zeigt die Sequenzierungsergebnisse für einige weitere Peptide mit natürlich vorkommenden Modifikationen. Alle Peptide aus Tabelle 47 weisen als Gemeinsamkeit eine terminale Modifizierung auf. Dabei konnte, außer für den Fall des pyro-Glutamats, keines der untersuchten Peptide mit einer N- α -Modifizierung durch eine der verwendeten unspezifischen Aminopeptidasen sequenziert werden. Auf C-terminaler Seite dagegen ist ein freier C-Terminus keine notwendige Voraussetzung für Sequenzierbarkeit. Die untersuchten, C-terminal amidierten Peptide lassen sich sowohl mit CPY als auch CPP gut abbauen.

Tabelle 47: Sequenzresultate weiterer modifizierter Peptide

Bezeichnung	Sequenzresultat	erfolgreiche Sequenzierung erfolgte mit
N α -Ac-P7-919	Ac-ALELFRK	-
N α -Ac-P7-808	Ac-GRDGFSK	-
N α -Ac-P14-1421	Ac-HGTVVLTALGGILK	-
N α -Ac-P12-1204	Ac-GASSGPTIEEVD	-
Substance P	RPKPQQFFGLM-NH ₂	CPY: CPY ₁ / CPY ₂
P12-1360-NH ₂	PTSFGYDKPHVL-NH ₂	CPY: CPY ₁ / CPY ₂ CPP: CPP ₁ / CPP ₂ sb(CP-I): CPY+CPP / CPP+CPY
P12-1365-NH ₂	VVHSGYRHQVPS-NH ₂	CPY ₂ CPP ₂ sb(CP-I): CPY+CPP / CPP+CPY
Neurotensin	(pyro) ELYENKPRRPYIL	APM ₂

3.5 Einfluss der MALDI-Matrix und Suppressionseffekte

Im Bezug auf die MALDI-MS Messung der enzymatisch generierten Peptidleitern wurden auch mögliche Suppressionseffekte näher untersucht. Unter dem Begriff der „Suppression“ soll dabei im Bezug auf die MALDI-MS die Tatsache verstanden werden, dass die Signalintensität bei der Messung eines bestimmten Peptids in Anwesenheit anderer Komponenten gegenüber einer Messung des reinen Peptids erniedrigt ist. Als „andere Komponenten“ sind insbesondere Salze aber auch andere Peptide in Betracht zu ziehen.

Als Ausgangspunkt der Untersuchungen wurden in den Suppressionsstudien zunächst mit CHCA als Matrix und UV-MALDI-MS gearbeitet. Bereits Versuche an binären Peptidmischungen zeigten für diese Kombination, dass ein 2 bis 10-facher Überschuss eines Peptids gegenüber dem anderen Peptid dessen Signalintensität um 50% erniedrigen kann [Kratzer_1 1998]. Vergleicht man die Signalintensitäten äquimolarer Mischungen von Peptiden der Dynorphinleiter (Dyn-6 bis Dyn-17; Tabelle 13), so stellt man gegenüber den Signalintensitäten bei der Messung der einzelnen Peptide eine noch deutlichere Suppression der Signale in der Mischung fest (gleiche Peptidmenge eines Peptids in der Mischung und bei der Einzelmessung). Die Signalintensität des Peptid in der Mischung kann hier um bis zu 20 mal geringer sein als die Signalintensität bei der Einzelmessung für das betreffende Peptid. In der Reihe der Dynorphin-Peptide zeigte sich der Suppressionseffekt bei den kürzeren Leiterpeptiden (z.B. Dyn-6 oder Dyn-7) am intensivsten. Die Unterdrückungseffekte, die bei der Messung von Leiterpeptiden auftraten, konnten analog auch bei anderen, ganz willkürlichen zusammengestellten Peptidmischungen beobachtet werden. Gerade die für die Untersuchung von Peptiden mittels MALDI-MS, im allgemeinen verwendete Kombination aus UV-Anregung und CHCA-Matrix zeigt besonders starke Suppression für Peptide in allen analysierten Gemischen und deutliche Nachteile gegenüber anderen getesteten Kombinationen. Aus den Ergebnissen für die Kombination aus CHCA und UV-MALDI-MS kann auf eine Abhängigkeit der beobachteten Suppression von der Basizität und Hydrophobizität der Peptide geschlossen werden. Die hydrophoben Eigenschaften von Peptid und Matrix wirken sich über einen *pre-desorption* Effekt auf die beobachtete Suppression aus. Hierbei bestimmt das Zusammenspiel aus Matrix- und Peptidhydrophobizität die unterschiedlichen Einbaugeschwindigkeiten der verschiedenen Peptide in die Gitterstruktur der kristallisierenden Matrix. Je nachdem wie gut der Einbau der einzelnen Peptide in die Matrix beim Kristallisationsvorgang erfolgen kann, tritt Suppression für bestimmte Peptide mehr oder weniger in Erscheinung [Kratzer_1 1998]. Als weitere Ursache für eine Suppression lässt sich das Konkurrieren der Peptide um eine begrenzte Anzahl von Ladungsträgern diskutieren. Dieser Effekt, der erst nach der Erzeugung von Molekülen in der

Gasphase zum Tragen kommt, kann als *post-desorption* Effekt bezeichnet werden. Hier spielt, nach den Resultaten für die Dynorphinleiter zu urteilen, die Peptidbasizität eine wichtige Rolle. Insbesondere argininhaltige Peptide zeigen einen stark unterdrückenden Effekt gegenüber anderen Peptiden. Aufgrund der weitaus geringeren Gasphasenbasizität von Lysin und Histidin, werden Peptide, die nur diese basischen Aminosäuren besitzen, weitaus stärker unterdrückt als argininhaltige Peptide. Die Abspaltung eines Arginins dürfte für den genannten *post-desorption* Effekt eine weitaus wichtigere Rolle zu spielen als die Abspaltung eines Lysins oder Histidins.

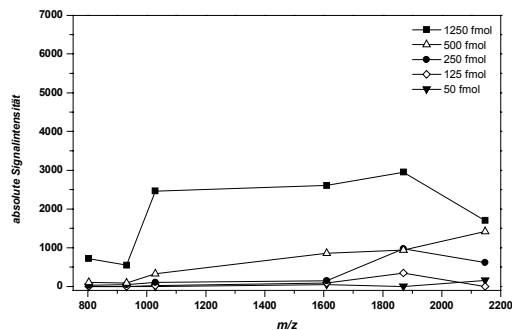
Die Resultate zeigen letztendlich, dass Suppressionseffekte mit dem Standard-MALDI-System „CHCA / UV-MALDI-MS“ zu einem nicht zu unterschätzenden Empfindlichkeitsproblem bei der Leitersequenzierung führen können. Insbesondere wenn, bedingt durch eine ungünstige Abspaltungskinetik, einige Leiterpeptide nur in geringen Mengen auftreten, können solche Suppressionseffekte auch die Entstehung von Sequenzlücken im Leiterspektrum begünstigen.

Im Hinblick auf eine Minimierung der geschilderten Suppressionseffekte wurden mit DHB und DHBs weitere Matrices in Kombination mit UV-MALDI-MS untersucht. Zusätzlich wurde mit Bernsteinsäure als Matrix ein System in Kombination mit IR-MALDI-MS untersucht. Abbildung 57 stellt die absoluten Signalintensitäten verschiedener Dynorphinleiterpeptide dar, wie sie bei Messung mit unterschiedlichen Matrices und Laserwellenlängen erhalten wurden. Für alle Spektren wurde die Intensität auf 25 Schuss normiert. Eine Quantifizierung über MALDI-MS ist zwar prinzipiell problematisch, in den dargestellten Fällen wurde jedoch über eine sehr große Anzahl von Laserimpulsen bei einer gleichzeitig sehr hohen Anzahl beschossener Matrixstellen je Probe summiert (200-250 Schuss für Messungen mit UV-Laser). Mit diesem Aufwand war es möglich die Abweichung bei Wiederholungsmessungen in einem Bereich um 20% zu bringen. Die beobachteten Unterschiede in Abbildung 57 sind somit weitgehend signifikant.

In den in Abbildung 57 gezeigten Grafiken steht jeder gezeigte Datenpunkt für ein bestimmtes Dynorphin-Peptid gemäß seiner Masse. Die einzelnen Datenreihen geben die Intensitäten der Dynorphin-Peptide für eine aufgetragene Peptidemengen wider (im Gemisch: Stoffmenge jedes einzelnen Peptids im Gemisch). In der linken Spalte von Abbildung 57 sind die Intensitäten dargestellt, wie sie sich bei der Einzelmessung für die reinen Dynorphine ergeben. Die Grafiken in der rechten Spalte ergeben sich für die Dynorphin-Peptide bei Messung äquimolarer Mischungen. Beim Vergleich der Grafiken ist zu beachten, dass die Intensitätskala aus Übersichtsgründen für die Mischungen nur 50% der Skala für die Einzelmessungen beträgt.

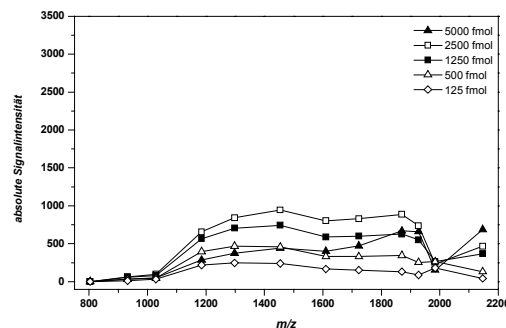
Abbildung 57: Signalintensitäten verschiedener Dynorphinleiterpeptide bei Messung der einzelnen Peptide (links) und bei Messung der Peptide in äquimolaren Gemischen (rechts)

Messung der einzelnen Leiterpeptide

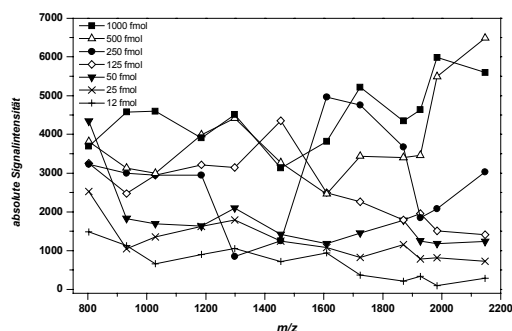


MALDI-Laserwellenlänge: UV 337nm
Matrix: CHCA

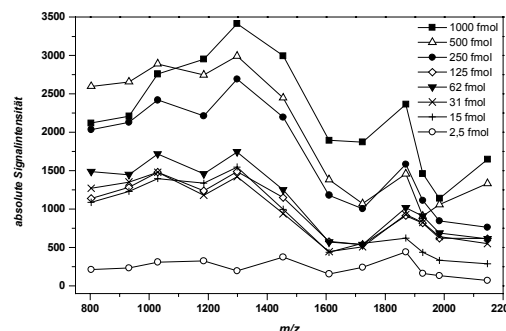
Messung äquimolarer Gemische der Leiterpeptide



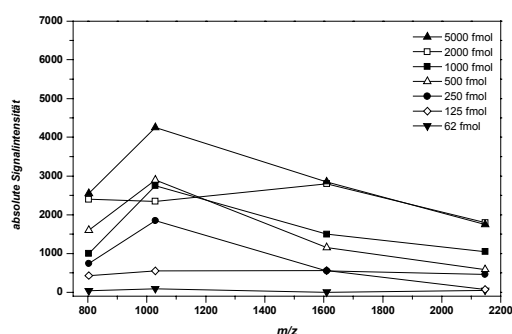
MALDI-Laserwellenlänge: UV 337nm
Matrix: CHCA



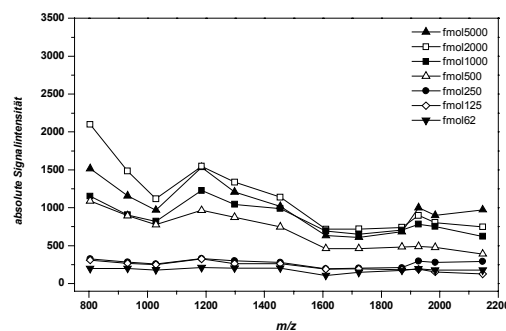
MALDI-Laserwellenlänge: UV 337nm
Matrix: DHB (analog DHBs)



MALDI-Laserwellenlänge: UV 337nm
Matrix: DHB (analog DHBs)



MALDI-Laserwellenlänge: IR 2,94nm
Matrix: Bernsteinsäure



MALDI-Laserwellenlänge: IR 2,94nm
Matrix: Bernsteinsäure

DHBs und DHB liefern sehr ähnliche Ergebnisse, allerdings zeigt DHBs etwas schlechtere Ergebnisse bei der Messung der Einzelpeptide als DHB.

Abbildung 57 zeigt, dass bis auf die Messreihen einzelner Peptide mit DHB als Matrix, die beobachteten Intensitäten generell mit der Peptidmenge abnehmen. Allerdings kann festgestellt werden, dass bei der Messung eines einzelnen, reinen Peptides das Signal im Vergleich

zur Messung dieses Peptids aus einer äquimolaren Mischung heraus im Mittel etwa die doppelte Intensität aufweist. Demnach ist eine Suppression unter den gegebenen Bedingungen für alle untersuchten Systeme festzustellen. Entscheidend für die Wahl eines Systems (Kombination Matrix / Laserwellenlänge) für die Anwendung bei der Leitersequenzierung war daher, welches System das geringste Ausmaß an Suppression aufweist. Diesbezüglich liefert das Standardsystem CHCA / UV-MALDI-MS die schlechtesten Resultate. Besonders die hydrophileren und weniger basischen Dynorphine Dyn-6 mit Dyn-8 (von Dyn-9 auf Dyn-8 wird N-terminal das Arginin abgespalten !) lassen sich in Leiterpeptidmischungen mit dieser Kombination kaum nachweisen. Für die hydrophileren, weniger basischen Dynorphine schneiden die restlichen Systeme, insbesondere jedoch die Kombination DHB / UV-MALDI-MS am besten ab. Auch bezüglich der Empfindlichkeit zeigt dieses System die besten Resultate. Bis in den unteren Femtomol-Bereich können alle Leiterpeptide aus der äquimolaren Mischung heraus noch gut detektiert werden.

Unter den realen Bedingungen einer Leitersequenzierung, muss auch in einem gepufferten System eine möglichst geringe Suppression gewährleistet sein. Abbildung 58 zeigt die Resultate für Suppressionen in drei UV-MALDI-MS Systemen bei Anwendung von CHCA, DHB und DHBs in Gegenwart graduell zunehmender Mengen an Natriumcitrat. Das Salz wird in Form einer wässrigen Lösung (1,0 µl) nach dem Trocknen der Probe aufgetragen und simuliert so bei der nachfolgenden MALDI-Messung den Salzgehalt der Enzymlösung.

Beim Vergleich der Darstellungen ist zu berücksichtigen, dass der bei DHB und DHBs gezeigte Bereich der Ordinatenachsen viermal größer ist, als für CHCA. Die Tabellen unter Punkt 2.5.2 (S. 84) zeigen, dass die angewendeten Natriumcitratkonzentrationen im Realfall bei der Leitersequenzierung im Bereich von 0,1-13mM liegen. Die für die Suppression untersuchten Citratkonzentrationen spiegeln also die Verhältnisse in der Praxis gut wider.

Die bereits ohne Salz niedrigen Intensitäten der Peptidsignale für das CHCA-System nehmen bereits bei Anwesenheit geringer Citratkonzentrationen ab 1mM noch einmal um 25-50% ab. Ganz im Gegensatz dazu zeigt sich bei DHB-Matrix eine sehr hohe Salztoleranz bei der MALDI-MS. Die Signalintensität lässt erst bei 10mM Natriumcitrat für einzelne Peptide – vor allem für die größeren Dynorphine – eine sichtbare Suppression erkennen. Auch bei DHBs kann mit steigender Citratkonzentration erst bei 10mM Natriumcitrat eine deutliche Unterdrückung einzelner Leiterpeptide festgestellt werden. Im Mittel sind absolute Signalintensitäten für Peptidleitern, die mit DHB und DHBs gemessen wurden vergleichbar. Beide Matrices sind nach den vorliegenden Resultaten der CHCA-Matrix im Hinblick auf Empfindlichkeit und das Auftreten von Suppressionseffekten auf jeden Fall vorzuziehen.

Abbildung 58: Suppressioneffekte bei Dynorphinleiterpeptiden in Abhängigkeit der Matrix bei UV-MALDI-MS in Anwesenheit unterschiedlicher Mengen Natriumcitrat

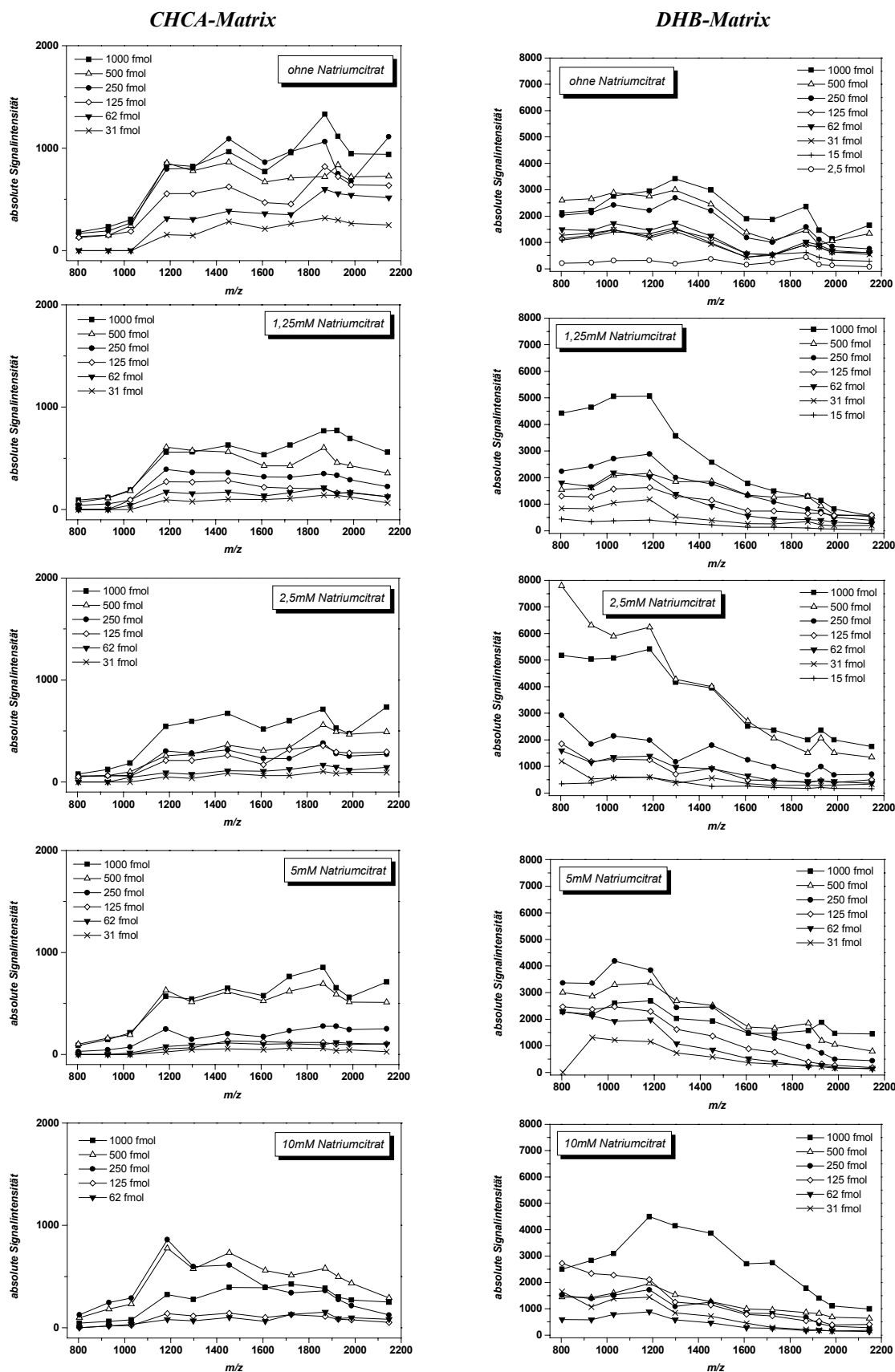
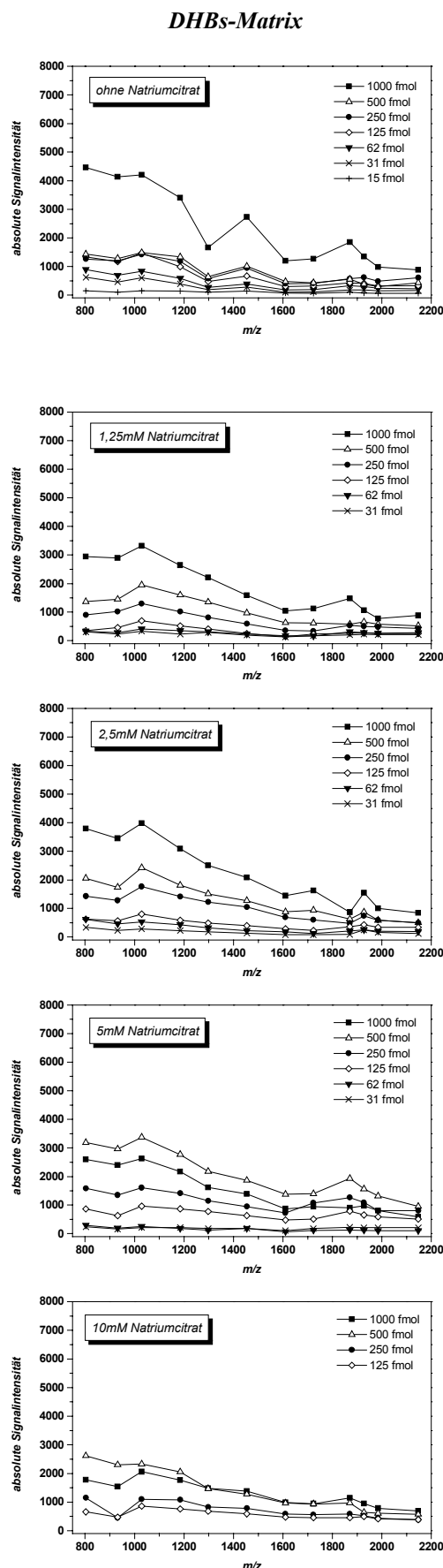


Abbildung 58: Fortsetzung



Auch ein Großteil der in dieser Arbeit sequenzierten Peptide wurde sowohl mit CHCA, als auch mit DHB gemessen. In nahezu allen Fällen lieferte hier eine Messung mit DHB die besseren Resultate. „Besser“ heißt in diesem Kontext: mit DHB als Matrix konnte bei der gleichen enzymatischen Sequenzierung im Gegensatz zu CHCA entweder überhaupt eine Peptidleiter beobachtet werden oder die Peptidleiter war bei Verwendung von DHB einfacher (höhere Signalintensitäten) beziehungsweise weiter (mehr Leiterpeptide sichtbar) auszulesen.

Als hydrophile Matrices zeigen DHB und DHBs eine geringfügig erhöhte Signalunterdrückung für hydrophobere Peptide, allerdings nur innerhalb des jeweiligen Matrixsystems. Die Suppressionsresultate für hydrophobe Peptide bei DHB und DHBs in Abbildung 57 und Abbildung 58 zeigen nämlich im Vergleich zu CHCA zumeist trotzdem bessere oder ähnliche Signalintensitäten. DHBs Matrix wurde im Rahmen dieser Arbeit nicht näher untersucht. Ebenso konnte die Kombination „Bernsteinsäure / IR-MALDI-MS“ aufgrund technischer Probleme mit dem Infrarotlaser-system, welches sich noch in der Entwicklungsphase befindet, nicht eingehender untersucht werden. Die IR-MALDI-MS, die ihr Hauptanwendungsgebiet eigentlich in der Proteinanalytik finden soll, bietet nach den ersten Resultaten in dieser Arbeit (siehe Abbildung 57) auch im Hinblick auf die Peptidanalytik ein gutes Potential.

Eine der CHCA vergleichbare Abhängigkeit der Signalintensität von Peptidbasizität oder – hydrophobizität kann für die Kombination „Bernsteinsäure / IR-MALDI-MS“ nicht bestätigt werden. Die Verteilung der Intensitäten über alle Leiterpeptide ist sehr regelmäßig. Das Fehlen von Abhängigkeiten der genannten Art, lässt sich bei der Anregung mittels IR mit hoher Wahrscheinlichkeit auf den deutlich erhöhten Probenabtrag beim Laserbeschuss zurückführen. Die bei der Desorption mittels UV-Beschuss bekannten Oberflächeneffekte werden dadurch weitgehend vermieden oder zumindest stark abgeschwächt.

Die Resultate lassen somit in der Summe lediglich DHB oder die Variante DHBs (DHB + HMBS) in Kombination mit der UV-MALDI-MS als geeignete Matrices für eine empfindliche Methode zur Leitersequenzierung erscheinen.

3.6 Sequenzierung auf der PVDF-Membran

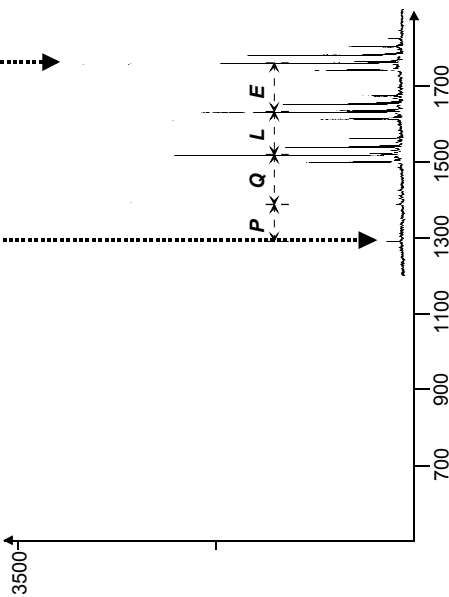
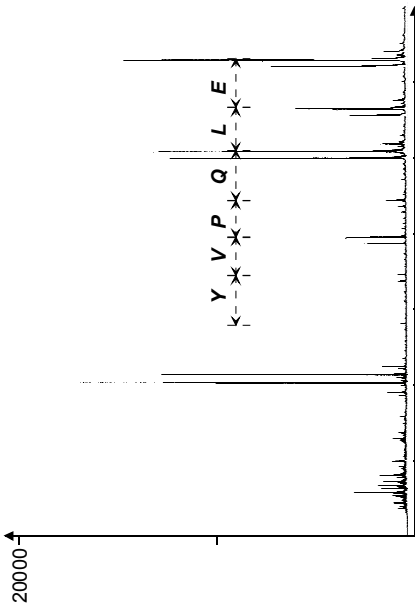
Enzymatische Sequenzierungen von gespotteten Peptiden auf Polyvinylidenfluorid (PVDF) Membranen konnten auf C-terminaler Seite mit CPY ebenfalls durchgeführt werden. Als Voraussetzung für eine erfolgreiche Abspaltung war dabei nach Aufbringen der Probe auf die Membran ein zusätzlicher Benetzungsschritt mit Methanol direkt vor der Enzymaufgabe. Offensichtlich fungiert das Methanol als „Vermittler“ zwischen der wässrigen Enzymlösung und der hydrophoben PVDF-Membran. Abbildung 59 stellt die Resultate der C-terminalen Sequenzierungen mit CPY auf dem MALDI-Target (Trägersubstrat: Metall – Edelstahl) und auf einer Immobilon PSQ Membran (Trägersubstrat: PVDF) für drei Peptide aus einer Spaltung mit Endoproteinase GluC (Phosphatpuffer) von α -Casein gegenüber.

Wie zu erkennen ist, wurden in allen Fällen bei der Sequenzierung auf der Membran ebenfalls Leitersequenzen einer Länge von mindestens vier Aminosäuren erhalten. Alle Spektren in Abbildung 59 wurden mit CHCA-Matrix aufgenommen und die Desorption erfolgt mittels UV-Laser (337nm). Für die MALDI-Präparation der Proben auf der Membran stellt sich die Verwendung einer CHCA Matrix-Lösung in Toluol/Methanol 50/50 (v/v) + 0,1% TFA bei einer Konzentration von 20mg/ml als optimal heraus.

Abbildung 59 (folgende Seite): MALDI-MS Leiterspektren von drei verschiedenen GluC-Fragmenten des α -Caseins nach Sequenzierung mit CPY; Präparation mit CHCA; Desorption von Metall und Membran mittels UV-Laser

C-terminale Sequenzierung von
 α -Casein GluC-Fragment 112-125
QLLRKKYKVPQLE (1756,885 Da)

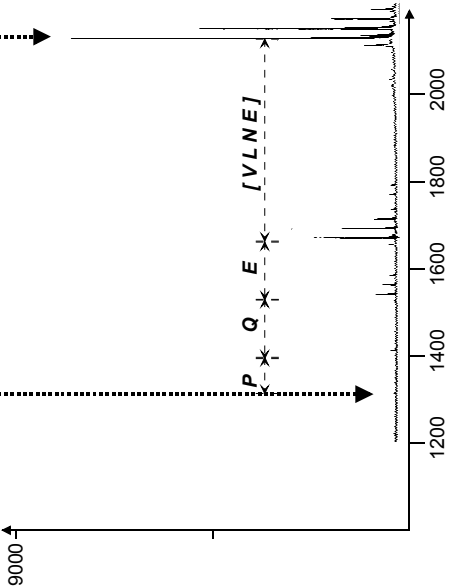
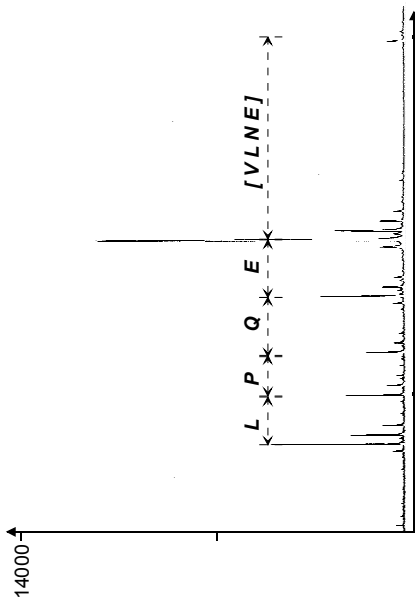
Sequenzierung auf dem Metall-Target



Sequenzierung auf der PVDF-Membran

C-terminale Sequenzierung von
 α -Casein GluC-Fragment 16-33
RPKHPIKHQGLPQEVLE (2120,172 Da)

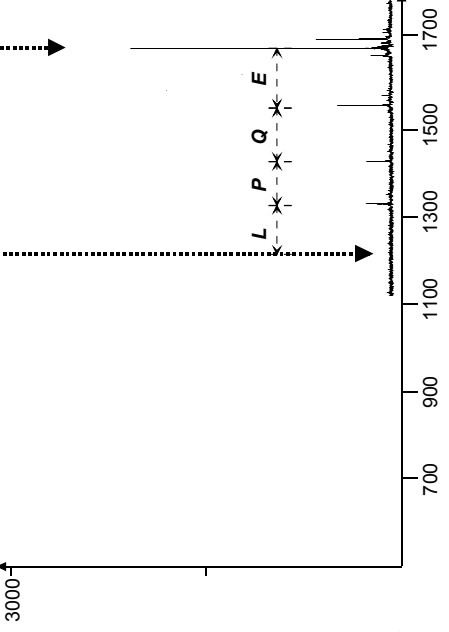
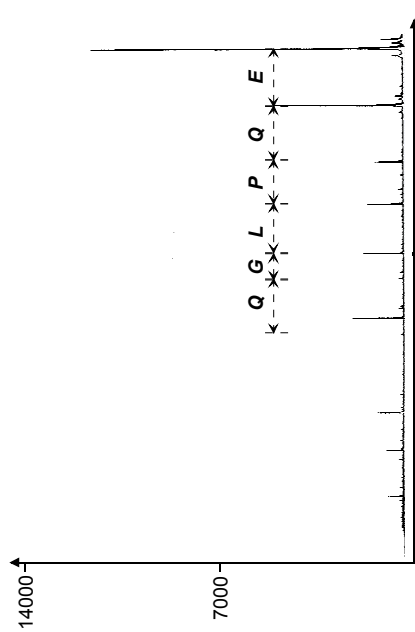
Sequenzierung auf dem Metall-Target



Sequenzierung auf der PVDF-Membran

C-terminale Sequenzierung von
 α -Casein GluC-Fragment 16-29
RPKHPIKHQGLPQE (1664,934 Da)

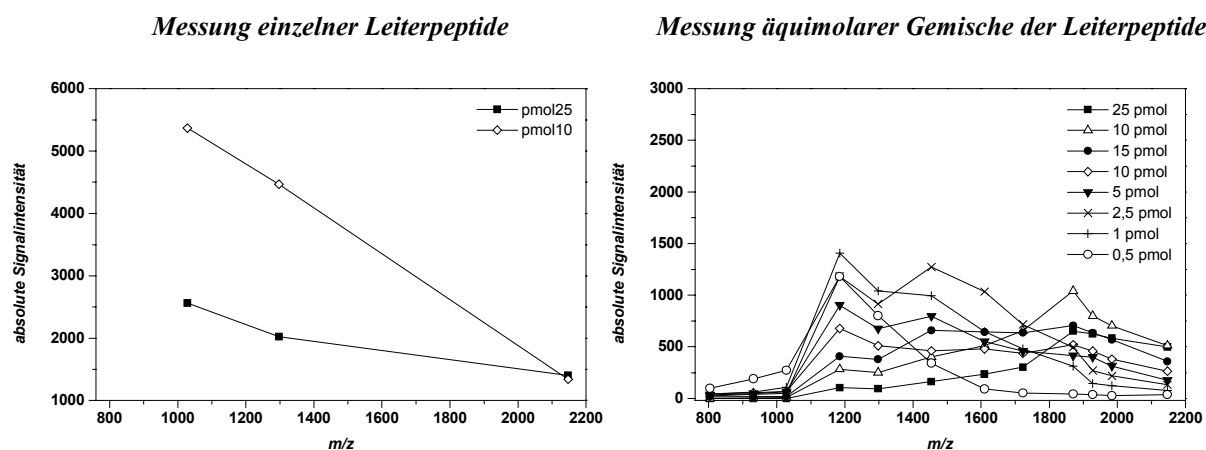
Sequenzierung auf dem Metall-Target



Sequenzierung auf der PVDF-Membran

Der Grund für die Verwendung von CHCA-Matrix bei der Leitersequenzierung auf der Membran lag in den hier zumeist besseren Signalintensitäten im Vergleich zu DHB-präparierten Proben auf der Membran. Die entsprechenden DHB-Präparationen liefern auf der PVDF-Membran keine vergleichbar guten Ergebnisse wie auf dem Metall-Tartget. Die Untersuchung von Suppressionseffekten mit dem System „CHCA / UV-MALDI-MS“ bei Messung von Immobilon PSQ Membran ist in Abbildung 60 dargestellt.

Abbildung 60: Signalintensitäten verschiedener Dynorphinleiterpeptide auf Immobilon PSQ bei Messung der einzelnen Peptide (links) und bei Messung der Peptide in äquimolaren Gemischen.



Wie die Messung der Einzelpeptide Dyn-8, Dyn-10 und Dyn-17, sowie die Messung der vollständigen Dynorphinleitern Dyn-6 bis Dyn-17 bei verschiedenen Konzentrationen zeigt, sind die Intensitäten denen der entsprechenden Präparationen und Messungen auf dem Metall-Target durchaus vergleichbar (vergleiche Abbildung 58). Im Hinblick auf die angestrebte Anwendung der „Membrantechnik“ im Bereich der Mikrosequenzierung (Leitersequenzierung gespotteter HPLC-Fractionen auf PVDF-Membranen; siehe Theorieteil, Abschnitt 1.3.1) war die Empfindlichkeit im Vergleich zu den entsprechenden *on-target* Sequenzierungen (Trägersubstrat: Metall – Edelstahl) etwas geringer, wie eine Betrachtung der Signalintensitäten in Abbildung 59 zeigt. Es wurde jeweils 1,0 μ l der gleichen Fraktion bei *on-target* und Membransequenzierung aufgetragen und sequenziert. Die Signalintensitäten der Ausgangs-peptide in den Leiterspektren liegen bei den Membranmessungen um einen Faktor 2-6 niedriger als bei der Messungen vom Metall-Target. Bei den nachfolgenden Leiterpeptiden sind die Intensitätsunterschiede dann noch etwas stärker ausgeprägt (Intensität bei Leiterpeptiden der Membranmessungen 3-6 mal geringer). Die vergleichenden Intensitätsdaten nach Sequenzierung auf der Membran und auf dem Metall-Target deuten auch auf einem schlechteren

Peptidabbau auf der Membran hin. Unterstützt wird diese These durch die Tatsache, dass für die vergleichbaren Sequenzierungsergebnisse auf Membran und Metall-Target wie sie in Abbildung 59 dargestellt sind, bei der Sequenzierung auf der Membran der Enzymbedarf (= aufgetragene Enzymmenge) um den Faktor 10 höher liegt. Während die Sequenzierungen auf dem Metall-Target mit nur 4 ng CPY (1,2 mU) durchgeführt wurden, waren es bei den Sequenzierungen auf der PVDF-Membran 40 ng CPY (12 mU). Mit 4 ng CPY konnte auf der Membran kein Abbau beobachtet werden. Der höhere Bedarf an Enzym ist mit Sicherheit teilweise dadurch zu begründen, dass ein Teil des Enzym ebenfalls an die Membran bindet und somit nicht mehr für den Peptidabbau zur Verfügung steht.

Allgemein zeigen die Leiterpeptidspektren in Abbildung 59 auch einen weitergehenden Abbau der Peptide bei Sequenzierung auf dem Metall-Target. Im Vergleich zur Sequenzierung auf der Membran wurden hier in der Regel um 1-2 Aminosäuren mehr abgebaut, trotz der Tatsache, dass die eingesetzte Enzymmenge bei Sequenzierung auf der Membran 10 mal größer war.

4 Diskussion und Bewertung

4.1 Experimentelle Bedingungen

Die Experimente zeigen, dass eine umfassende N- bzw. C-terminale Sequenzierungsmethode, die auf möglichst alle Peptidsequenzen anwendbar ist, über die Verwendung einer einzigen Amino- bzw. Carboxypeptidase nicht möglich ist. Aufgrund der in der Literatur beschriebenen Spaltungsspezifitäten der verwendeten Exopeptidasen (Tabelle 2 / Tabelle 3, S. 16) und der Tatsache, dass hier eine absolut unspezifische Peptidase weder für die N- noch für die C-terminale Aminosäureabspaltung zur Verfügung steht, entspricht dieses Ergebnis sicherlich den theoretischen Erwartungen. Aber auch unter den getesteten Enzymkombinationen erweist sich keine als absolut universell anwendbar, um jedes Sequenzierungsproblem zu lösen. Eine Gesamtbetrachtung zeigt, dass für die meisten Peptide die besten Sequenzierungsergebnisse durch Anwendung von bis zu zwei verschiedenen Enzymen oder Enzymkombinationen erhalten wurden. Um mit dieser Strategie wiederum bei erfolgreicher Sequenzierung einen möglichst langen Sequenzabschnitt zu erhalten, ist es dann erforderlich, in einer zeit- oder konzentrationsabhängigen Spaltung zwei oder drei weitere Experimente durchzuführen. Somit war bereits zu Beginn der Arbeit die Vorgabe eines einfachen Satzes von Sequenzierungsparametern bezüglich Enzymmenge, Spaltungszeit, Spaltungstemperatur, pH-Wert und Pufferbedingungen ist unter diesen Voraussetzungen nicht zu erwarten. In den folgenden Abschnitten sollen jedoch einige grundlegende Rahmenbedingungen, die sich aus entsprechend erfolgreichen Sequenzierungsergebnissen ergeben haben, vorgegeben werden.

4.1.1 Optimale Spaltungstemperaturen und Spaltungszeiten

Die Parameter Spaltungstemperatur und Spaltungszeit sind bei der *on-target* Sequenzierung primär aneinander gekoppelt, da die Zeit bis zum Eintrocknen der Enzymlösung und somit die eigentliche Sequenzierungszeit in dem gegebenen, offenen System natürlich von der Temperatur abhängt. Dabei ist die Spaltungszeit weiterhin vom aufgetragenen Enzymvolumen abhängig. Die Auftragung eines Volumens von 1 µl Enzymlösung erweist sich für die Sequenzierungen bei der gegebenen Spotgröße von 2 mm auf dem MALDI-Target als Standard gut geeignet. Im weiteren basieren daher alle Aussagen auf der Auftragung dieses Standardvolumens.

Bezüglich der Spaltungstemperatur stellt sich ein Bereich von 25-35°C als geeignet heraus. Die meisten Versuche, insbesondere auch später bei der automatisierten Sequenzierung, wurden bei 30°C durchgeführt. Eine reproduzierbare und vor allem kontrollierte Sequenzierung ist im angegebenen Temperaturbereich von 25-35°C für das aufgetragene Standard-enzymvolumen gut möglich. Unter den üblichen Umgebungsbedingungen (Luftfeuchtigkeit) beträgt die Zeit bis zum Eintrocknen der Enzymlösung (= Spaltungszeit) ca. 4-6min. Dieser Zeitrahmen erlaubt, neben der üblichen Steuerung der Reaktionskinetik über die Enzymmenge, auch zusätzlich eine gute Kontrolle der Kinetik über die Reaktionszeit. Da die Abspaltungskinetik im Normalfall über die Variation der aufgetragenen Enzymmenge und somit die angewandte Enzymaktivität erfolgt, sollte eine Veränderung der Reaktionszeit prinzipiell nicht erforderlich sein. Aus den meisten Sequenzierungsergebnissen der eingesetzten Amino- und Carboxypeptidasen, sowie der Peptidasmischungen erweist sich im Bereich der getesteten Enzymaktivitäten auch hier im Normalfall eine Reaktionszeit von 4-6min als zweckmäßig. In einigen Einzelfällen, in denen gerade die Abspaltung der ersten Aminosäuren zu schnell ablief, konnte aber nur über die Verkürzung der Reaktionszeit auf 1-2min eine Optimierung erreicht werden, nicht über die Verringerung der Enzymmenge. Auch konnte zum Teil zusätzliche Sequenzinformation nicht über eine einmalige Erhöhung der aufgetragenen Enzymmenge erhalten werden, sondern nur durch eine Verlängerung der Reaktionszeit. Diese Verlängerung ließ sich sowohl über nachträgliche die Auftragung von Wasser auf den bereits enzymbehandelten Probenspot erreichen, als auch über eine wiederholte Enzymauftragung. Wird die Reaktionszeit ausschließlich durch Auftragen von Wasser verlängert, so ist es auch wahrscheinlich, dass das Enzym mit fortlaufender Reaktionszeit an Aktivität verliert. Andererseits wird bei mehrmaliger Auftragung von Enzymlösung natürlich nicht nur die Reaktionszeit verlängert, sondern es findet auch eine Beschleunigung der Aminosäureabspaltung über die Enzymmenge statt. Die wiederholte Enzymaufgabe zeigte folgerichtig in der Regel auch bessere Sequenzierungsergebnisse als die Verlängerung der Reaktionszeit über nachträgliche die Auftragung von Wasser. Bei der mehrmaligen Auftragung der Enzymlösung ergaben sich jedoch durch die gleichzeitige Akkumulation von Puffersalzen bei der anschließenden MALDI-MS Messung auch Probleme mit Suppressionseffekten. Solche Suppressionseffekte führten vor allem im Zusammenhang mit der Anwesenheit von Citratpuffer aus den Carboxypeptidasen CPY und CPP und Ammoniumsulfat aus APM zu einem allgemeinen Rückgang der Signalintensitäten im Leiterspektrum, weshalb eine mehr als zweimalige Auftragung der Lösungen dieser Enzyme mit den gegebenen Enzympräparationen vermieden werden sollte. Wie Tabelle 34 und Tabelle 35 (S. 146ff.) zeigen, wurden bezüglich einer Verlängerung der Reaktionszeit

Untersuchungen mit der Aminopeptidase APM und den Carboxypeptidasen CPY und CPP durchgeführt. Entscheidend für die Interpretation der Ergebnisse in den genannten Tabellen ist die Tatsache, dass die Sequenzinformation, die aus den verlängerten Reaktionszeiten resultiert, zumeist komplementär zu den maximalen Sequenzinformationen bei kurzen Reaktionszeiten von 4-6 min sind. Reaktionszeiten bis zu ca. maximal 12min sind in dieser Hinsicht optimal. Darüber hinaus konnte bei noch längeren Reaktionszeiten kein Vorteil durch zusätzliche Sequenzinformation erzielt werden.

Im Fall der Anwendung einer seriell binären Enzymkombinationen zeigt sich, dass hier allgemein eine Reaktionszeit von 10-12 min die besten Ergebnisse liefert. Dabei sollte das zweite Enzym erst nach dem Eintrocknen der ersten Enzymlösung aufgetragen werden.

4.1.2 Enzyme, Enzymkombinationen und Enzymmengen

In Tabelle 48 sind die Bereiche optimaler Mengen der wichtigsten Enzyme und Enzymkombinationen für die N- und C-terminale Leitersequenzierung angegeben. Dabei handelt es sich um Auswahl derjenigen Enzyme und Enzymkombinationen aus Tabelle 15 bis **Fehler! Verweisquelle konnte nicht gefunden werden.** die bei den durchgeführten Leitersequenzierungen zu den besten Resultaten geführt haben. Die bevorzugten Enzymmengen der hier nicht eigens aufgeführten Enzymkombinationen liegen im angegebenen Optimumbereich der betreffenden Einzelenzyme.

Tabelle 48: Empirisch ermittelte, optimale Enzymaktivitäten (Enzymmengen) für die Anwendung bei der enzymatischen Leitersequenzierung

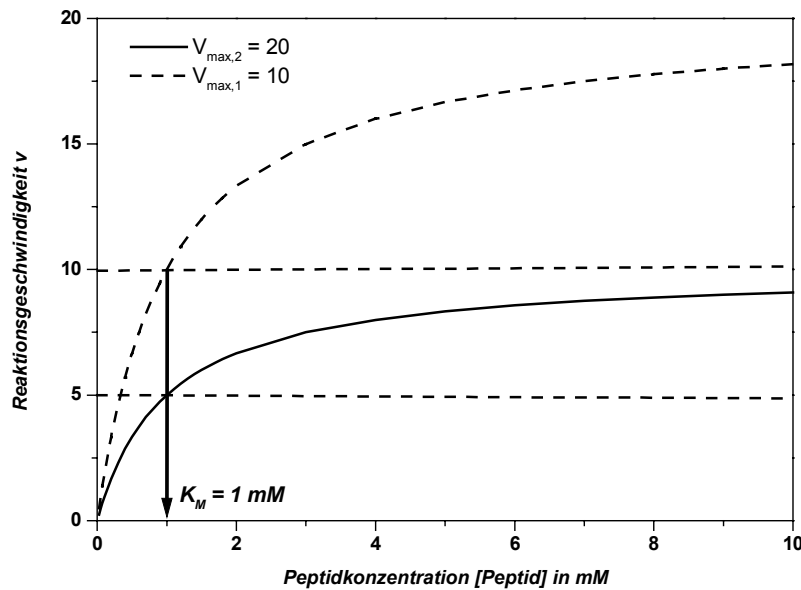
Enzym / Enzymkombination	optimaler Aktivitätsbereich [mU]	Enzymmenge ¹ [ng]
<i>Aminopeptidasen</i>		
APM	0,4 – 2	100 – 500
AAP	5 – 25	40 – 200
API	4 – 19	6 – 30
<i>Carboxypeptidasen</i>		
CPY	0,3 – 6	1 – 20
CPP	0,05 – 1	2 – 40
CPW	1 – 5	25 – 100
CPA	1 – 2	30 – 60
CPB	2 – 4	10 – 30
sb(CP-I)	CPY: 0,3 – 6 / CPP: 0,05 – 1	CPY: 1 – 20 / CPP: 2 – 40
pb(CP)	CPY: 0,3 – 6 / CPP: 0,05 – 1	CPY: 1 – 20 / CPP: 2 – 40

¹ Aufgetragenen Enzymmenge pro Target-Spot in 1µl Enzymlösung

Die angegebenen Enzymmengen in Tabelle 48 sind dabei natürlich im Zusammenhang mit den spezifischen Aktivitäten der verwendeten Enzympräparationen zu sehen (Tabelle 9 / Tabelle 11, S. 67). Bei einer Steuerung der Abspaltungskinetik zeigt sich, dass bereits 2-3 Spaltungsansätze mit unterschiedlichen Enzymmengen für die verschiedenen Enzyme vollkommen ausreichen, um die maximale Sequenzinformation zu erhalten. Eine noch feinere Abstufung der Enzymmengen erweist sich als nicht notwendig. Eine solche Abstufung ist auch nicht zweckmäßig, um kinetische Abspaltungsunterschiede an besonders kritischen Stellen in der Sequenz zu nivellieren. Bei extrem ungünstiger Abspaltungskinetik einzelner Aminosäuren in der Sequenz, die dazu führt, dass Lücken im Leiterspektrum auftreten, sollte besser ein anderes Enzym oder eine andere Enzymkombination zur Sequenzierung angewendet werden, um den fehlenden Sequenzabschnitt zu erhalten.

Die Dynamik der Methode, d.h. die Anwendbarkeit einer Kombination aus enzymatischer Spaltung und MALDI-MS erscheint nach den Resultaten grundsätzlich vom mittleren Pikomol-Bereich (ca. 50pmol) bis in den unteren Femtomol-Bereich (ca. 10fmol) gegeben. Bei größeren Stoffmengen stellt zumeist die Dynamik der MALDI-MS durch Sättigung der verwendeten MCP-Detektoren einen limitierenden Faktor dar, weshalb eine Verdünnung der Probenlösung notwendig ist. Weiterhin wird oft eine Suppression der kürzeren Leiterpeptide durch die große Menge an Ausgangspeptid beobachtet. Folglich erlaubt die Methode prinzipiell ohne zusätzliche Verdünnungsschritte, Peptidstoffmengen über einen Bereich von maximal vier Größenordnungen zu sequenzieren und anschließend auch zu detektieren. Die Angabe eines allgemein gültigen, engen Bereichs der anzuwendenden Enzymmengen erscheint im Zusammenhang mit der Sequenzierung von Realproben zunächst nur bedingt möglich, denn es ist zu berücksichtigen, dass die zu sequenzierenden Peptidmengen zumeist unbekannt sind. Eine Betrachtung der zum Teil stark unterschiedlichen K_M -Werte der verwendeten Enzyme in BRENDA [Schomburg 2000] für verschiedene Substrate und Spaltungsbedingungen (pH, Ionenstärke und Temperatur) zeigt jedoch, dass für den genannten dynamischen Bereich die Sequenzierungen im allgemeinen nicht bei Substratsättigung ausgeführt wurden. Abbildung 61 veranschaulicht an einem Beispiel (Enzym mit $k_M = 1\text{mM}$), für zwei verschiedenen Enzymkonzentrationen (resultierend in $V_{max,1}$ und $V_{max,2}$), dass im linearen Bereich mit $[Peptid] < k_M$ eine Anpassung der Enzymmenge an die Substratmenge nicht erforderlich ist. Die Reaktionsgeschwindigkeit v steigt hier proportional mit der zu sequenzierenden Peptidmenge an. Die Dynamik der Methode ist also bei konstanter Enzymmenge auch für stark unterschiedliche Substratmengen gegeben.

Abbildung 61: Reaktionsgeschwindigkeit v in Abhängigkeit der Substratkonzentration für ein Enzym mit $k_M=1\text{mM}$ berechnet für zwei unterschiedliche Enzymmengen ($V_{\max,1} = 10$ und $V_{\max,2} = 20$)



Der Bereich der Substratsättigung ist für die Leitersequenzierung dagegen ungeeignet, da bei maximaler Reaktionsgeschwindigkeit und mit steigender Substratkonzentration dann der Peak des Ausgangspeptids das Spektrum dominiert, was zur Suppression kleinerer Leiterpeptide führen kann. Gleichung (13) zeigt, dass außerhalb der Substratsättigung vor allem der die katalytische Aktivität beschreibende Quotient k_{cat}/K_M die Angabe eines universellen Bereichs für die einzusetzende Enzymmenge erschwert.

$$v = \frac{k_{cat}}{k_M} \cdot [Peptid] \cdot [E_{gesamt}] \quad (13)$$

wobei gilt: $[Peptid] \ll k_M$

k_{cat} wird als sogenannte *Turnover-Zahl* oder Wechselzahl bezeichnet und entspricht im Michaelis-Menten Modell k_2 aus Gleichung (7). (Weitere Bezeichnungen siehe S. 45ff.)

$$k_{cat} = \frac{V_{\max}}{[E_{gesamt}]} \quad (14)$$

Der Quotient k_{cat}/K_M ist unter anderem stark von der Art des Substrats abhängig. Hier stellen nicht nur unterschiedliche Proteinfragmente aus der Endopeptidasespaltung unterschiedliche Substrate dar. Auch im Lauf der Exopeptidase-Sequenzierung entstehen über die

Veränderung des Peptidterminus ständig neue Substrate. Tabelle 49 zeigt einige Beispiele für k_{cat}/K_M Werte der Hydrolyse von Dipeptidsubstraten mit der Carboxypeptidase CPY und der Aminopeptidase LAP. Diese kinetischen Daten zeigen ebenfalls noch einmal deutlich den wichtigen Einfluss der sekundären Substratspezifität der über die Aminosäure in nachfolgender Position Xbb zur abzuspaltenden Aminosäure Xaa ausgeübt wird. Für die grau unterlegten Felder in Tabelle 49 zeigt der Abbau des Phenylalanins bei unterschiedlichen Aminosäuren in der Position Xbb k_{cat}/K_M Werte, die sich um einen Faktor von bis zu 10^3 unterscheiden !

Tabelle 49: Kinetische Daten für den N- und C-terminalen Abbau unterschiedlicher Substrate mit LAP und CPY

Substrat ¹	k_{cat}/K_M [mM ⁻¹ s ⁻¹]	Substrat ^{1,2}	k_{cat}/K_M [mM ⁻¹ s ⁻¹]
Daten für den N-terminalen Abbau mit LAP bei pH 8,3 und 25°C nach [Hanson 1976]		Daten für den C-terminalen Abbau mit CPY bei pH 6,5 im Natriumphosphatpuffer und 25°C nach [Hayashi 1977]	
H₂N-Ala-Phe-COOH	37	CBZ-Phe-Ala-COOH	210
H₂N-Phe-Phe-COOH	3833	CBZ-Phe-Phe-COOH	840
H₂N-Leu-Phe-COOH	4821	CBZ-Phe-Leu-COOH	1300
H₂N-Gly-Phe-COOH	0,5	CBZ-Phe-Gly-COOH	37
H₂N-Trp-Phe-COOH	1158	CBZ-Phe-Pro-COOH	34
		CBZ-Phe-Glu-COOH	5,8
		CBZ-Phe-His-COOH	3,6
H₂N-Phe-Gly-COOH	2,8	CBZ-Gly-Phe-COOH	1,3
H₂N-Phe-Tyr-COOH	1988	CBZ-Leu-Phe-COOH	470
H₂N-Phe-Trp-COOH	952	CBZ-Glu-Phe-COOH	4,0
		CBZ-His-Phe-COOH	7,3
H₂N-Tyr-Tyr-COOH	645	CBZ-His-Tyr-COOH	6,2
H₂N-Trp-Trp-COOH	7000	CBZ-Ala-Ala-COOH	21
		CBZ-Leu-Leu-COOH	101
		CBZ-Gly-Pro-COOH	kein Abbau

¹ Um die abgespaltene Aminosäure (C- oder N-terminal) besser kenntlich zu machen, wurden die freie Aminogruppe und der freie Carboxy-Terminus ebenfalls angegeben – die abgespaltene Aminosäure ist fett gedruckt.

² CBZ: Benzyloxycarbonyl

Letztendlich zeigt die Beurteilung der erhaltenen Teilsequenzen von allen Peptiden, dass sich bei der Sequenzierung eines Terminus mit verschiedenen Einzelenzymen und Enzymkombi-

nationen eine möglichst vollständige Sequenzinformation erst aus den kombinierten Resultaten der Anwendung mehrerer Einzelpeptidasen und / oder Peptidasekombinationen ergibt.

Möchte man dabei einzelne Peptidasen oder Peptidasekombination als besonders erfolgversprechend hervorheben, so zeigen CPY in der Anwendung als Einzelpeptidase, sowie die binären Kombinationen sb(CP-I) und pb(CP) auf C-terminaler und APM auf N-terminaler Seite überdurchschnittlich gute Sequenzierungsergebnisse. Für die genannten Peptidasen und Peptidasekombinationen war einerseits der Anteil sequenzliefernder Fragmente größer als bei den anderen getesteten Enzymen, andererseits wurden hier auch häufiger längere Teilsequenzen mit mehr als 10 Aminosäuren erhalten. Solche längeren Sequenzen sind sowohl für eine erfolgreiche Homologiesuche, als auch für die Herstellung von Oligonucleotidsonden besser geeignet (siehe Kapitel 4.6 Sequenzlängen und Anwendungsbereich).

4.1.3 pH-Bedingungen

Die Betrachtung optimaler pH-Bereiche für die Leitersequenzierung muss einerseits getrennt nach C- und N-terminaler Sequenzierung erfolgen, andererseits getrennt nach der Anwendung einzelner Peptidasen oder Peptidasekombinationen. Da das generelle Ziel bei der Leitersequenzierung nicht allein eine möglichst schnelle, sondern eine schnelle und gleichzeitig kontrollierte Abspaltung der Aminosäuren ist, dienen die in der Literatur angegebenen pH-Optima der Exopeptidasen lediglich als Anhaltspunkt für die ausgeführten Experimente. Für die kontrollierte Abspaltung erwies es sich in den Versuchen oft als zweckmäßig, bewusst leicht außerhalb des pH-Optimums einer Exopeptidase zu arbeiten. In der Praxis hat sich gezeigt, dass die experimentelle Durchführung durch die Tatsache erleichtert wird, dass bei allen Exopeptidasen zumeist eine größere Toleranz für die pH-Bedingungen gegeben ist. So lassen sich beispielsweise für die C-terminale Sequenzierung verschiedener Peptide mit unterschiedlichen Carboxypeptidasen bei Veränderung des pH-Wertes sicherlich Intensitätsänderungen der einzelnen Leiterpeptide beobachten. Konkret trat jedoch innerhalb eines Intervalls von circa zwei pH-Einheiten um einen optimalen pH-Wert für die Sequenzierung nur selten ein Verlust an Sequenzinformation ein. Für die C-terminale Sequenzierung wurden bezüglich dieser pH-Toleranz Beispiele in Abbildung 28 (S. 111) für (CPP und CPW) beziehungsweise in Abbildung 29 (S. 112) für (CPB und CPY) typische Beispiele aufgezeigt. In gleicher Weise ist eine Toleranz von ein bis zwei pH-Einheiten bei der Sequenzierung mit Aminopeptidasen gegeben.

Für die getesteten Exopeptidasen sind in Tabelle 50 die empirisch aus den Sequenzierungen ermittelten pH-Optima (Bereiche) für die Leitersequenzierung zusammengefasst.

Tabelle 50: Empirisch ermittelte pH-Optima der getesteten Exopeptidasen für die Leitersequenzierung

Enzym	optimaler pH-Bereich für die Leitersequenzierung	verwendeter Puffer
Aminopeptidasen		
APM	7 – 8	Tris 5 – 50mM
AAP	7 – 8	
API	7 – 8	
LAP	8 – 9	
Carboxypeptidasen		
CPY	5 – 7	pH < 7: Natriumcitrat 0,5 – 5mM pH > 7: Tris 5 – 50mM
CPP	4 – 6	
CPW	4,5 – 7,5 ¹	
CPA	6 – 9,5 ²	
CPB	6 – 8	

¹ besser im sauren pH-Bereich zwischen pH 4,5 und 6,0

² theoretischer Wert aus [Peterson 1976]

Die Werte in Tabelle 50 können allerdings auch nur als generelle Anhaltspunkte dienen. Unter Punkt 1.1.3 (S. 22) wurde am Beispiel der CPY gezeigt, dass sich optimale pH-Werte für die Abspaltung unterschiedlicher Aminosäuren nennenswert unterscheiden können. Diese Aussage wird durch die Experimente dieser Arbeit bestätigt. Die Tatsache, dass eine exakte Einhaltung eines definierten pH-Wertes keine notwendige Voraussetzung für eine erfolgreiche Sequenzierung darstellt, ist somit eine wichtige Grundlage der enzymatischen Leitersequenzierung. Die Toleranz der Leitersequenzierung bezüglich gewisser pH-Unterschiede macht die Methode nicht nur robust, sondern ist auch eine wichtige Voraussetzung für die einfache Anwendung serieller und insbesondere paralleler Peptidasekombinationen.

Wie Tabelle 50 zeigt, überschneiden sich die optimalen pH-Bereiche der Aminopeptidasen sehr gut, so dass kombinierte Anwendungen verschiedener Aminopeptidasen sowohl aufeinanderfolgend (sequenziell), als auch miteinander (als Mischung) in einfacher Weise möglich sind. Alle getesteten Aminopeptidasen, außer LAP, zeigen mit einem auf pH 7,3 eingestellten Tris-HCl Puffer für die Sequenzierung ein gutes Abspaltungsverhalten. Dementsprechend fanden APM, API und AAP auch in Peptidasekombinationen bei diesem pH-Wert eine erfolgreiche Anwendung.

Bei den Carboxypeptidasen unterscheiden sich die optimalen pH-Bereiche stärker. Als Einzelpeptidasen (nicht in Mischung) wurden CPA und CPB bei pH 7,5 im Tris-HCl Puffer angewendet. Als Standard bei CPY- und CPP-Sequenzierungen wurde dagegen im sauren pH-Bereich gearbeitet. Die Sequenzierung der Peptide mit CPY fand in den meisten Fällen erfolgreich bei pH 6,0 statt. Der optimale pH-Wert für die meisten CPP-Sequenzierungen lag bei 5,0 und für CPW-Sequenzierungen bei 4,0. Damit erscheint eine kombinierte Anwendung der Carboxypeptidasen CPA und CPB mit CPY, CPP oder CPW zunächst schwierig.

Im Hinblick auf die Sequenzierung von Spaltfragmenten aus Trypsinspaltungen und von Fragmenten aus Spaltungen mit Endoproteinase LysC wurden vor allem Kombinationen der unspezifischen Peptidasen CPY und CPP mit CPB optimiert. Da bei diesen Spaltpeptiden das durch die CPB abzuspaltende Lys oder Arg direkt am C-Terminus sitzt, wurden Kombinationen mit CPB sinnvollerweise für die serielle Enzymauftragung optimiert. Zuerst erfolgt der Abbau der C-terminal basischen Aminosäuren mit CPB bei pH 7,5 in Tris-HCl. Danach wird 1 µl einer wässrigen, 0,1%igen TFA Lösung aufgetragen. Die Überführung des überschüssigen Tris (pK_a 8,1) in die korrespondierende Säure schafft pH-Bedingungen, die auch mit CPY und CPP kompatibel sind. Nach dem Trocknen der TFA-Lösung auf dem MALDI-Target wird mit der unspezifischen Peptidase die Sequenzierung fortgesetzt. Die Abspaltung der basischen Aminosäuren (bis auf His) im ersten Schritt dieser Enzymkombination öffnet zudem die Möglichkeit für die nachfolgenden Sequenzierungsschritte den pH-Wert etwas saurer und damit besser für die Abspaltung nicht-basischer Aminosäuren einzustellen. Die in Abbildung 28 und Abbildung 29 dargestellten Ergebnisse zeigen jedoch, dass prinzipiell auch eine Anwendung der CPB mit den übrigen Peptidasen in einer Mischung möglich ist, nämlich in einem schmalen Bereich um pH 6.

Die kombinierte Anwendung von CPY, CPP und CPW ist dagegen aufgrund besser überlappender Aktivitätsbereiche generell in Form einer Mischung möglich. Die für die Leitersequenzierung experimentell bestimmten, optimalen pH Bereiche für CPY (pH 5-7) und CPP (pH 4-6,5) entsprechen weitgehend den in der Literatur angegebenen Aktivitätsbereiche dieser Carboxypeptidasen: pH 5,5-7,5 für CPY [Hayashi 1972] und pH 2,5-6,5 für CPP [Yokoyama 1975]. Für die Anwendung der CPW ergab sich im Bezug auf die Leitersequenzierung ein optimaler pH-Bereich von pH 4,5-7,5 (Abbildung 28). Ein entsprechender Vergleichswert aus der Literatur für den Aktivitätsbereich der CPW liegt nicht vor. Für die genannten, unspezifische Carboxypeptidase CPY, CPP und CPW ist somit eine kombinierte (parallele) Anwendung in einem relativ breiten pH-Fenster von pH 5,5-6,5 möglich. Für die sehr häufig eingesetzte Peptidasekombinationen sb(CP-I) (= CPY + CPP oder CPP + CPY), sowie pb(CP)

(= CPY / CPP) erweist sich in der Praxis ein pH-Wert von 5,0-5,5 als optimal. Grund für die Wahl eines etwas niedrigeren pH-Bereichs ist die Tatsache, dass Leiterspektren, die bei der Sequenzierung mit CPP bei pH-Werten über 5,5 registriert wurden, teilweise einen geringfügigen Rückgang der Sequenzinformation zeigen. Die Toleranz gegenüber pH-Veränderungen ist bei CPP im stärker sauren Bereich größer. Dagegen ergeben sich bei der Sequenzierung mit CPY auch bei pH 5,0 noch sehr gute Resultate, die gegenüber den Resultaten bei pH 6,0 nur selten Informationsverlust aufweisen.

Für die sequenzielle Anwendung der Peptidasen CPY und CPP konnte auf eine Umpufferung vor Zugabe der zweiten Peptidase – vergleichbar der Sequenzierung mit CPB – verzichtet werden. Je nach Molarität der Puffer der Einzelpeptidasen stellt sich hier ein für beide Peptidasen akzeptabler pH-Wert um 5,5 nach Auftragung der zweiten Peptidase ein.

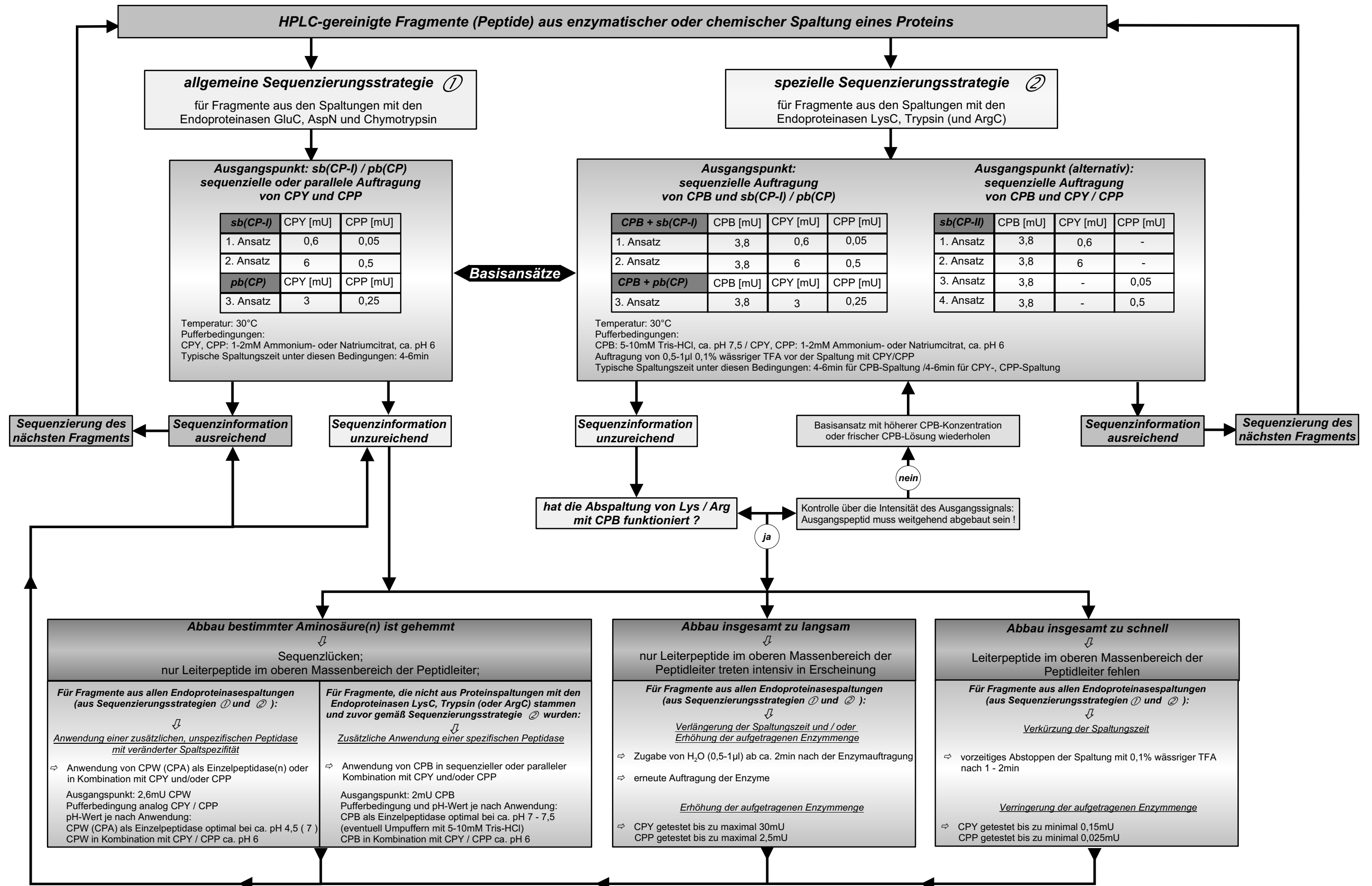
Auch Kombinationen von CPY und / oder CPP mit CPW zeigen bei pH-Werten um 5,0 ein besseres (schnelleres) Abspaltungsverhalten.

4.1.4 Praktische Sequenzierungsstrategien

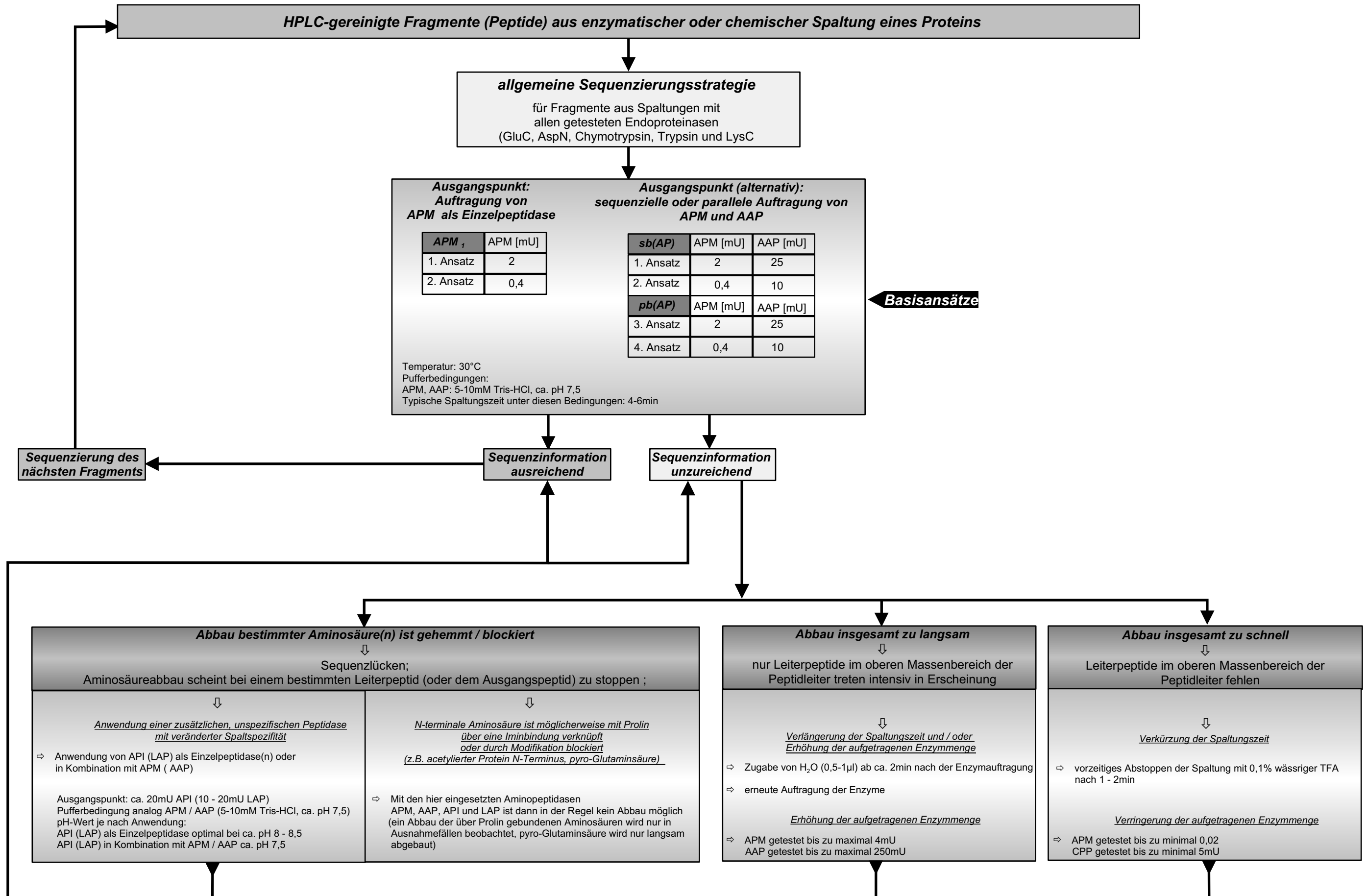
Auf den folgenden beiden Doppelseiten sind grundlegenden Strategien für eine C-terminale beziehungsweise N-terminale Leitersequenzierung zusammengefasst. Die hier vorgeschlagenen, experimentellen Vorgehensweisen konzentrieren sich in ihrem Anwendungsbereich in erster Linie auf die Sequenzierung von enzymatisch erzeugten Proteinfragmenten mit den in dieser Arbeit verwendeten Endoproteinasen. Außerdem wurden in den Schemata nur tatsächlich getestete Exopeptidasen berücksichtigt. Diejenigen Peptidasen und Peptidasekombinationen, die gemäß den experimentell erhaltenen Resultaten die besten Aussichten auf Erfolg zeigten (hoher Anteil von Proteinfragmenten, die Sequenzinformation liefern, Maximum der Sequenzlängenverteilung bei längeren Sequenzen) bilden für die Sequenzierungsstrategien bei C- und N-terminaler Sequenzierung sogenannte Basisansätze. Jede Probe wird also zunächst aufgeteilt und gemäß den genannten Basisansätze behandelt. Für die Sequenzierung derjenigen Peptide, für die diese Basisansätze keine oder nur unzureichende Sequenzinformation liefern, sind unterschiedliche Lösungsansätze der Problembehandlung dargestellt. Welche der gezeigten Möglichkeiten zur Modifikation der Sequenzierungsparameter verwendet werden sollte, muss dabei anhand der Leiterspektren, die mit den Basisansätzen erhalten wurden, individuell beurteilt werden.

Für die C-terminale Sequenzierung wird eine optimale Sequenzierungsstrategie im speziellen von der vorangegangenen Endoproteinasespaltung des Proteins abhängig gemacht. Wie die

Schema einer C-terminalen Sequenzierungsstrategie



Schema einer N-terminalen Sequenzierungsstrategie



Resultate zeigen, bringt bei der C-terminalen Sequenzierung von Fragmenten aus Spaltungen mit den Endoproteinasen LysC und Trypsin die zusätzliche Anwendung von CPB in den Basisansatz eine deutliche Verbesserung (in der Regel größere Sequenzlängen). Den grundlegenden Ausgangspunkt für die Basisansätze bei der C-terminalen Sequenzierung bilden jedoch Kombinationen der unspezifischen Peptidasen CPY und CPP. Bei Fragmenten aus Proteinspaltungen mit den Endoproteinasen LysC und Trypsin sollten diese unspezifischen Peptidasekombinationen nach einem Spaltungsschritt mit CPB erfolgen. Für den Fall, dass mit den angegebenen Basisansätzen keine ausreichende Sequenzinformation erzielt werden kann oder die Spektren keine eindeutige Interpretation erlauben, sind drei verschiedene Lösungsansätze aufgezeigt. Eine Kombination der verschiedenen Lösungsansätze oder auch eine rekursive Anwendung der Schritte ermöglicht unter Umständen eine weitergehende Optimierung. Möchte man eine unspezifische Peptidase zusätzlich zu CPP und CPY einsetzen, so wird in der Praxis zumeist nur CPW eine echte Ergänzung darstellen. In der Folge sollte praktisch nach dem Basisansatz die Durchführung von drei weiteren Ansätzen erfolgen: CPW (als Einzelpeptidase), CPW + CPY (als Mischung), sowie CPP + CPP (als Mischung). CPA zeigte in den Experimenten zwar gute Eigenschaften bei der Abspaltung hydrophober Aminosäuren, jedoch im Vergleich zu CPY und CPP keine zusätzliche Spezifität für einzelne Aminosäuren oder komplexere Sequenzbereiche, die durch CPY oder CPP eventuell nur langsam hydrolysiert werden.

Für eine N-terminale Sequenzierungsstrategie stellt sich die Vorgehensweise weniger komplex dar. Alle hier eingesetzten Aminopeptidasen (APM, AAP, API und LAP) sind mehr oder weniger unspezifisch. Für die Basisansätze erscheinen nach den Resultaten zunächst zwei Sequenzierungsansätze mit APM optimal. Alternativ sind auch die angegebenen Kombinationen aus APM und AAP als Basisansätze geeignet. Sofern die Sequenzierungsergebnisse auf ein Problem mit der Spaltspezifität der eingesetzten Peptidasen hindeuten sollte im folgenden Schritt API in drei weiteren Ansätzen als Einzelpeptidase sowie in sequenzieller und paralleler Kombination mit APM getestet werden: API (als Einzelpeptidase), APM + API (sequenziell), sowie APM + API (als Mischung).

4.2 Empfindlichkeit und Dynamik

Um im Rahmen der Leitersequenzierung eine Empfindlichkeit angeben zu können, muss sicherlich zunächst der Begriff der „Empfindlichkeit“ im Bezug auf die Leitersequenzierung als analytische Methode definiert werden. Die Empfindlichkeit soll sich dabei sicherlich auf die notwendige Stoffmengemenge des Ausgangspeptids beziehen, die für eine „erfolgreiche“ Sequenzierung benötigt wird. Der Begriff „Empfindlichkeit“ suggeriert zunächst, dass ein enger Bereich existiert, innerhalb dessen bei einer Verringerung der Stoffmenge des zu sequenzierenden Ausgangspeptids die gesamte Sequenzinformation auf einmal verloren geht. Eine solche Eingrenzung, wie sie in analoger Weise bei vielen anderen Methoden gemacht werden kann, ist im Bezug auf die Leitersequenzierung sicherlich nicht sinnvoll. In den meisten Fällen beobachtet man über einen Bereich von zwei bis drei Größenordnungen der sequenzierten Ausgangsmenge eine sukzessive Abnahme der Sequenzinformation in den zugehörigen Leiterpeptidspektren, wie aus den Resultaten unter Punkt 3.1.4 hervorgeht (S. 102). Inwiefern die Sequenzierung dabei als „erfolgreich“ bezeichnet wird, muss von der Länge der erhaltenen Teilsequenz, die aus dem Leiterspektrum ausgelesen werden kann, abhängig gemacht werden. Nimmt, wie in den Beispielen unter 3.1.4 (Abbildung 24 mit Abbildung 27) die Sequenzinformation mit abnehmender Menge des sequenzierten Ausgangspeptids ab, so ist es sicherlich sinnvoll für die Beurteilung des Sequenzierungsergebnisses die erhaltene Sequenzlänge bei einer bestimmten Konzentration auf die maximal zu erwartende Sequenzlänge zu beziehen. Diese maximale Sequenzlänge entspricht dabei derjenigen Sequenzlänge, die man für Peptid erwarten kann, wenn für die Sequenzierung eine genügende Stoffmenge des Peptids eingesetzt wird. Die Sequenzierungsversuche für synthetische Peptide wurden normalerweise ausgehend von einer Peptidmenge im unteren Pikomol-Bereich (1-5pmol) bis oberen Femtomol-Bereich (500fmol) gestartet. Eine Erhöhung der Peptidmenge (bis ca. 50pmol) erbrachte nur in Einzelfällen mehr Sequenzinformation, bedingt durch intensivere Signale von Leiterpeptiden geringer Intensität. In den meisten Fällen jedoch (so auch in den gezeigten Beispielen unter 3.1.4) ließ sich die Sequenzinformation bei einer Erhöhung der Ausgangsmenge des zu sequenzierenden Peptids über 1pmol nicht weiter steigern. Viele Leitersequenzierungen zeigen sogar bei einer Sequenzierung von 1pmol bis 500fmol und Verwendung von DHB-Matrix noch eine Detektorsättigung für einzelne Leiterpeptide.

Zusätzlich ist der Bereich, über den die Abnahme der Sequenzinformation erfolgt individuell vom der Aminosäuresequenz des betrachteten Peptids abhängig. Im Gegensatz zu einer Sequenzierung, die wie die Edman-Sequenzierung auf eine sehr kleine Zahl von potentiellen Analyten begrenzt ist (zumeist nur 20 verschiedene Aminosäuren), ist die Vielfalt der Analyten bei der Leitersequenzierung durch die Detektion der Leiterpeptide weitaus größer (betrachtet man allein Leiterpeptide mit 5-40 Aminosäuren, so ergeben sich ca. 10^{52} potentielle Analytpeptide!). Drei Aspekte sind dabei im Zusammenhang mit der Aminosäuresequenz des zu sequenzierenden Peptids im Bezug auf die Empfindlichkeit von Bedeutung:

❖ Massenspektrometrische Response des sequenzierten Peptids:

Die massenspektrometrische Response bei der UV-MALDI-MS ist unter anderem sehr stark von der Aminosäuresequenz abhängig [Olumee 1995]. So wurden beispielsweise in Vorversuchen dieser Arbeit UV-MALDI-MS Messungen mit verschiedenen, willkürlich ausgewählten synthetischen Peptiden (8-14 Aminosäuren) durchgeführt. Bei gleichen Peptidmengen und identischen Messparametern unterscheiden sich die beobachteten Signalintensität des Peptids mit der maximalen Signalintensität und des Peptids mit der minimalen Signalintensität um zwei bis drei Zehnerpotenzen [Kratzer_2 1998]. Damit ist die erreichbare Empfindlichkeit für die Sequenzierung eines Peptids primär von der Signalintensität abhängig, die sich bei der alleinigen Messung dieses Peptids in verschiedenen Konzentrationen mit einer bestimmten Matrix ergeben. In Abbildung 22 stellen daher für die beispielhaft untersuchten Peptide ACTH (18-39) und P15-1812 5fmol sowie für P12-1303 und P10-1213 50fmol die absolute Mindestmenge an Ausgangspeptid für die Sequenzierung dar. Bei ACTH (18-39) wurden für eine Stoffmenge im unteren Femtomol-Bereich (10fmol) immerhin 70% der maximalen Sequenzinformation erhalten. Die nachfolgende Tabelle fasst die Resultate aus 3.1.4 zusammen. 100% Sequenzinformation entsprechen dabei der maximal erreichbaren Sequenzinformation für das Peptid (100%: schwarze Felder; 100-50% graue Felder, 50-0% weiße Felder).

Tabelle 51: Sequenzinformationen bei C-terminaler Sequenzierung unterschiedlicher Ausgangsmengen

sequenziertes Peptid	Sequenzinformation in Aminosäuren (AS) und % der maximal erreichbaren Sequenzinformation für das Peptid bei einer Sequenzierung von			
	5pmol	500fmol	50fmol	10fmol
ACTH (18-39)	15 AS (100%)	15 AS (100%)	11 AS (73%)	11 AS (73%)
P15-1812	6 AS (100%)	5 AS (83%)	5 AS (83%)	0 AS (0%)
P12-1303	6 AS (100%)	5 AS (83%)	2 AS (33%)	0 AS (0%)

Tabelle 52: Sequenzinformationen bei N-terminaler Sequenzierung unterschiedlicher Ausgangsmengen

sequenziertes Peptid	Sequenzinformation in Aminosäuren (AS) und % der maximal erreichbaren Sequenzinformation für das Peptid bei einer Sequenzierung von			
	5pmol	500fmol	50fmol	5fmol
P15-1812	8 AS (100%)	8 AS (100%)	7 AS (88%)	1 AS (13%)
P12-1303	5 AS (100%)	5 AS (100%)	4 AS (80%)	2 AS (40%)
P10-1213	5 AS (100%)	5 AS (100%)	3 AS (60%)	0 AS (0%)

Für die Beurteilung der Empfindlichkeit stellt sich damit also für jedes Peptid die Frage, ob die erhaltene Sequenzinformation ausreichend ist. Für die Identifizierung über Datenbanksuche hängt die benötigte Sequenzinformation (Sequenzlänge) u.a. von der Suchstrategie ab (siehe unten: 4.6 Sequenzlängen und Anwendungsbereich).

Bei Verwendung von DHB-Matrix ergab sich für die Sequenzierungen der meisten Peptide bei Ausgangsmengen zwischen 500fmol und 50fmol noch maximale Sequenzinformation. Je nach Peptid konnte diese Menge in einigen Fällen noch mal um einen Faktor 10 reduziert werden. Empfindlichkeiten im mittleren bis unteren Femtomol-Bereich (ca. 10-100fmol) sind damit durchaus als realistisch anzusehen, werden jedoch insbesondere beim Fehlen basischer Aminosäuren – insbesondere Arginin – nicht immer erreicht. Nur in Einzelfällen, bei sehr basischen Sequenzen (z.B. zwei oder mehr Arginine, beziehungsweise Arginin und Lysin in der Peptidsequenz), zeigen sich auch beim enzymatischen Abbau von Attomol-Mengen (>500amol) im MALDI-MS noch kurze, lesbare Peptidleitern. Bei derart geringen Peptidmengen können sich vor allem auch die unter 3.5 (S. 177) erwähnten Suppressionseffekte negativ auf die Empfindlichkeit auswirken.

❖ Abgespaltene Aminosäuren im Verlauf der Sequenzierung:

Zu berücksichtigen ist bei der Betrachtung der Empfindlichkeit auch die Tatsache, dass sich durch die Abspaltung einzelner Aminosäuren das Ionisierungsverhalten einiger Leiterpeptide stark vom Ionisierungsverhalten anderer Leiterpeptide unterscheiden kann. So kann beispielsweise die Abspaltung einzelner basischer Aminosäuren eines Peptids die Ionisierung der nachfolgenden Leiterpeptide deutlich verschlechtern. Besonders ausgeprägt zeigt sich dieser Effekt z.B. beim Übergang von Dyn-9 (RPKLKWDNQ) zu Dyn-8 (PKLKWDNQ) aus der Serie der Dynorphin-Leiterpeptide bei Abspaltung des Arginins. Äquimolare Mischungen der Dynorphin-Leiterpeptide im Bereich von 125fmol bis 15pmol zeigen für das Dyn-8 einen Abfall der Signalintensität um ca. eine Zehnerpotenz gegenüber Dyn-9. Für nachfolgende Leiterpeptide kann sich in vergleichbaren Fällen eine geringe Ausgangsmenge des zu

sequenzierenden Peptids sehr ungünstig auswirken. Da argininhaltige Peptide in der Regel eine sehr gute MS-Response besitzen sollte man berücksichtigen, dass sich das geschilderte Phänomen insbesondere bei der C-terminalen Sequenzierung von Peptiden aus Trypsin- und ArgC Spaltungen negativ auswirken wird. Basierend auf den gleichen Überlegungen sind N-terminale Sequenzierungen von Peptiden aus Trypsin- und ArgC Spaltungen im Bezug auf die Empfindlichkeit positiv zu bewerten, da ein potentielles Arginin (oder Lysin) auf jeden Fall in allen gebildeten Leiterpeptiden erhalten bleibt.

❖ Verwendete Matrix und Laserwellenlänge – Suppressioneffekte:

Wie die Resultate aus 3.5 (Einfluss der MALDI-Matrix und Suppressioneffekte) zeigen, hängt die MS-Response in der MALDI-MS nicht nur von der Aminosäuresequenz ab, sondern auch stark von der verwendeten Matrix und der zur Desorption verwendeten Laserwellenlänge. Betrachtet man die unter 3.5 untersuchten, äquimolaren Peptidgemische der Dynorphin-Leiter, so lassen sich für die vier Matrixsysteme in Kombination mit UV-Laserdesorption (337nm) und IR-Laserdesorption (2,94µm) deutliche Unterschiede in der Empfindlichkeit feststellen (Tabelle 53).

Tabelle 53: Vergleich der MALDI-MS Messung äquimolarer Dynorphinleitern (Dyn-7 bis Dyn-17; 11 Peptide) mit unterschiedlichen Matrixsystemen und Desorptionswellenlängen

Matrix	Laser (Desorptionswellenlänge)	Äquimolare Mischung bei deren MALDI-MS Messung noch alle Leiterpeptide im Spektrum sichtbar sind
Messungen auf dem Metall-Target		
CHCA	UV (337nm)	~ 100-150fmol / Peptid
DHB	UV (337nm)	~ 1fmol / Peptid
DHBs	UV (337nm)	~ 10fmol / Peptid
Bernsteinsäure	IR (2,94µm)	~ 50-100fmol / Peptid
Messungen auf PVDF-Membran (Immobilon PSQ)		
CHCA	UV (337nm)	~ 500fmol / Peptid

Die Messung mit DHB, die in diesem Fall die besten Resultate erbringt, zeigt auch bei der Messung der enzymatisch *on-target* generierten Peptidleitern die besten Empfindlichkeiten. Nicht zuletzt spielt hierbei sicherlich auch die unter 3.5 in Abbildung 58 gezeigte, im Vergleich zu den anderen Matrices, hohe Toleranz der DHB gegenüber den für die enzymatische Spaltung notwendigen Puffersalzen, eine entscheidende Rolle für die höhere Empfindlichkeit.

Auch wenn für die Sequenzierungen der proteolytischen Proteinfragmente nur wenige quantitative Resultate vorliegen, ergibt sich eine gute Übereinstimmung mit den bisher gezeigten

Resultaten synthetischer Peptide. Die für die Sequenzierung untersuchten Proteine wurden vor der Bearbeitung in lyophilisierter Form eingewogen und lagen damit zu Beginn der Aufarbeitung in bekannter Stoffmenge (und Konzentration) vor. Infolge der unbekannten Wiederfindungsraten der einzelnen Aufarbeitungsschritte sind jedoch die Konzentrationen der Proteinfragmente in den HPLC-Fraktionen und somit letztlich auch die sequenzierten Peptidmengen unbekannt. In einigen Fällen waren die Peptidmengen in den HPLC-Fraktionen ausreichend um eine Quantifizierung mittels Aminosäureanalyse (Ninhydrinmethode) durchzuführen. Aus der Spaltung mit Endoproteinase GluC (Phosphatpuffer) des α -Caseins konnten zwei Peptide quantifiziert werden, wobei sich folgende Peptidkonzentrationen der Fraktionen errechneten:

- ❖ Sequenzabschnitt 1-14: (PKHPIKHQLPQE): $2,6 \pm 0,8 \text{ pmol}/\mu\text{l}$
- ❖ Sequenzabschnitt 19-30 (NFFVAPFPE): $4,3 \pm 1,4 \text{ pmol}/\mu\text{l}$

Der ermittelte, durchschnittliche Peptidgehalt von $3,4 \pm 1,1 \text{ pmol}/\mu\text{l}$ dürfte jedoch an der oberen Grenze liegen, denn in allen anderen Fällen lagen die Aminosäuremengen bei der Quantifizierung unter der Nachweisgrenze. Viele HPLC-Fraktionen der Proteinfragmente mussten vor der Sequenzierung 1/10 mit ACN/Wasser 50/50 (v/v) + 0,1% TFA verdünnt werden, da die MALDI-MS Messung der unverdünnten Peptidleiter zur Detektorsättigung führte. Die Leitersequenzierung führt zwar auch bei größeren Peptidmengen (bis zu 50-100 pmol) zumeist nicht zu einem Verlust an Sequenzinformation, die erhaltenen Peptidleitern lassen sich jedoch oft deutlich schlechter aus den Massenspektren auslesen. Dies liegt zum einen an der Suppression durch das im Überschuss vorhandene Ausgangspeptid, zum anderen an den resultierenden Massenungenauigkeiten der Peptidsignale bei Detektorsättigung.

Für Verdünnungsreihen zeigt dagegen das Beispiel der Sequenzierung des Fragments 97-110 von α -Casein (QLRLKKYKVPQLE, Proteinspaltung mit Spaltung Endoproteinase GluC im Phosphatpuffer), dass bei folgenden Verdünnungen noch die gleiche Sequenzinformation aus dem Massenspektrum ausgelesen werden kann, wie für die unverdünnte Ausgangsfraktion:

- ❖ CPY-Sequenzierung: tolerierbare Verdünnung 1/50
⇒ **60 fmol** sequenziertes Peptid
(ca. 2-4 mM Natriumcitrat in der Präparation)
- ❖ APM-Sequenzierung: tolerierbare Verdünnung 1/10
⇒ **300 fmol** sequenziertes Peptid
(ca. 50 mM Tris und 250 nmol (!) Ammoniumsulfat in der Präparation)

Eine Sequenzierung von 300-60fmol Ausgangspeptid ohne Verlust an Sequenzinformation sind mit den Ergebnissen aus der Sequenzierung synthetischer Peptide konsistent.

Wie die Sequenzierung der Bradykininderivate (Abbildung 50; S. 168) zeigt, sind Empfindlichkeiten bis zu 10fmol auch bei der Sequenzierung von Peptidderivaten zu erreichen, selbst wenn die Sequenzierung des Rohprodukts erfolgt.

Die für die Empfindlichkeit diskutierten Aspekte beeinflussen in entsprechender Weise auch die Dynamik der Sequenzierungsmethode. Die obere Grenze der direkt – also ohne vorherige Verdünnung sequenzierbare Peptidmenge liegt im allgemeinen im mittleren Pikomol-Bereich (10-100pmol). Einerseits ist hier eine Limitierung über die Sättigung des Detektors gegeben, andererseits werden in der Praxis durch einzelne Leiterpeptide, die in sehr hohen Konzentrationen vorliegen, starke Suppressionseffekte beobachtet [Kratzer_1 1998, Kratzer_2 1998]. Die minimal notwendige Peptidmenge für die Sequenzierung liegt wie gesehen im Bereich zwischen 500fmol und 10fmol, wobei in den betrachteten Beispielen (Tabelle 51 und Tabelle 52) mehr als 70% der maximal erreichbaren Sequenzinformation als ausreichend für die Identifizierung angesehen wurden. Damit umfasst der dynamische Bereich der Methode in einer Größenordnung von ca. 10^3 bis 10^4 .

4.3 Spektreninterpretation

Zusätzlich zur reinen Übersetzung der Massendifferenzen der Leiterpeptide in die Aminosäuresequenz lassen sich aus einigen Massenspektren zusätzliche Informationen auslesen. Diese Informationen können die Interpretation der Leiterspektren entweder erleichtern oder auch Hinweise auf Aminosäuren geben, die noch gar nicht abgebaut wurden. In diesem Zusammenhang soll im folgenden kurz die Bedeutung von Oxidationen und metastabilen Fragmentierungen diskutiert werden. Außerdem soll gezeigt werden, dass über die gewählte Endopeptidasespaltung des Proteins indirekt auch eine bedingte Unterscheidung isobarer Aminosäuren möglich ist.

4.3.1 Oxidationen

Oxidationen der Aminosäuren Methionin und Tryptophan sind häufig zu beobachtende Modifikationen in den MALDI-Massenspektren von Peptiden. Die beobachteten Oxidationen sind hier eine Folge der Probenaufarbeitung in Anwesenheit von Sauerstoff. Mögliche Oxidations- und Folgeprodukte der Methionin- und Tryptophanoxidation sind nachfolgend dargestellt – zusätzlich sind die beobachteten Massenverschiebungen der Reaktionsprodukte angegeben.

Abbildung 62: Schematische Darstellung der Tryptophanoxidation

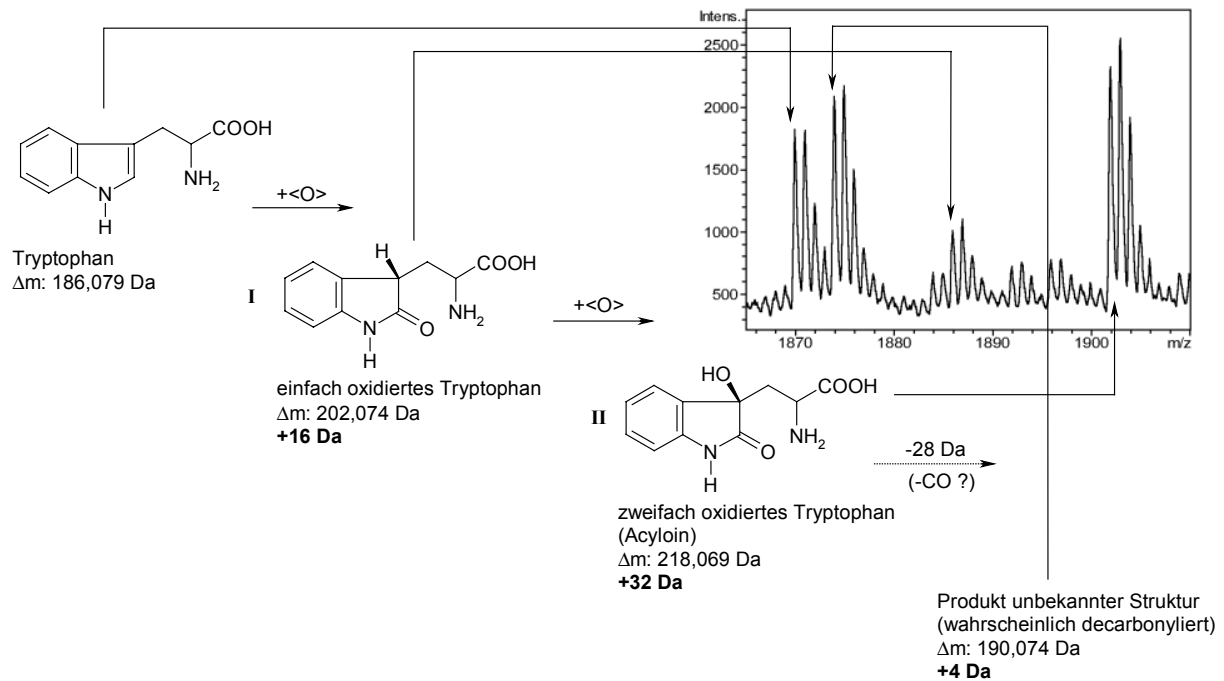
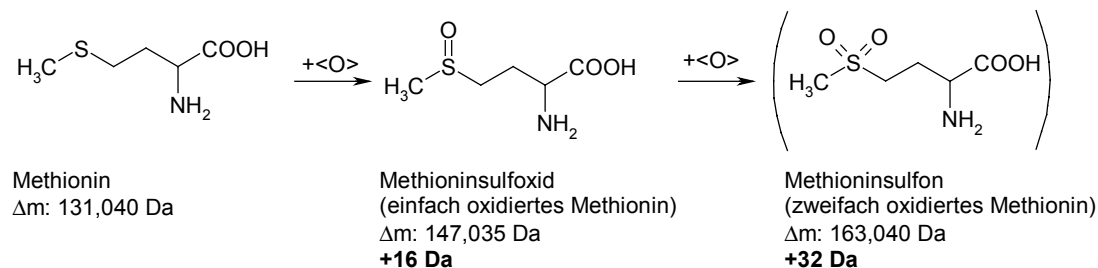


Abbildung 63: Schematische Darstellung der Methioninoxidation



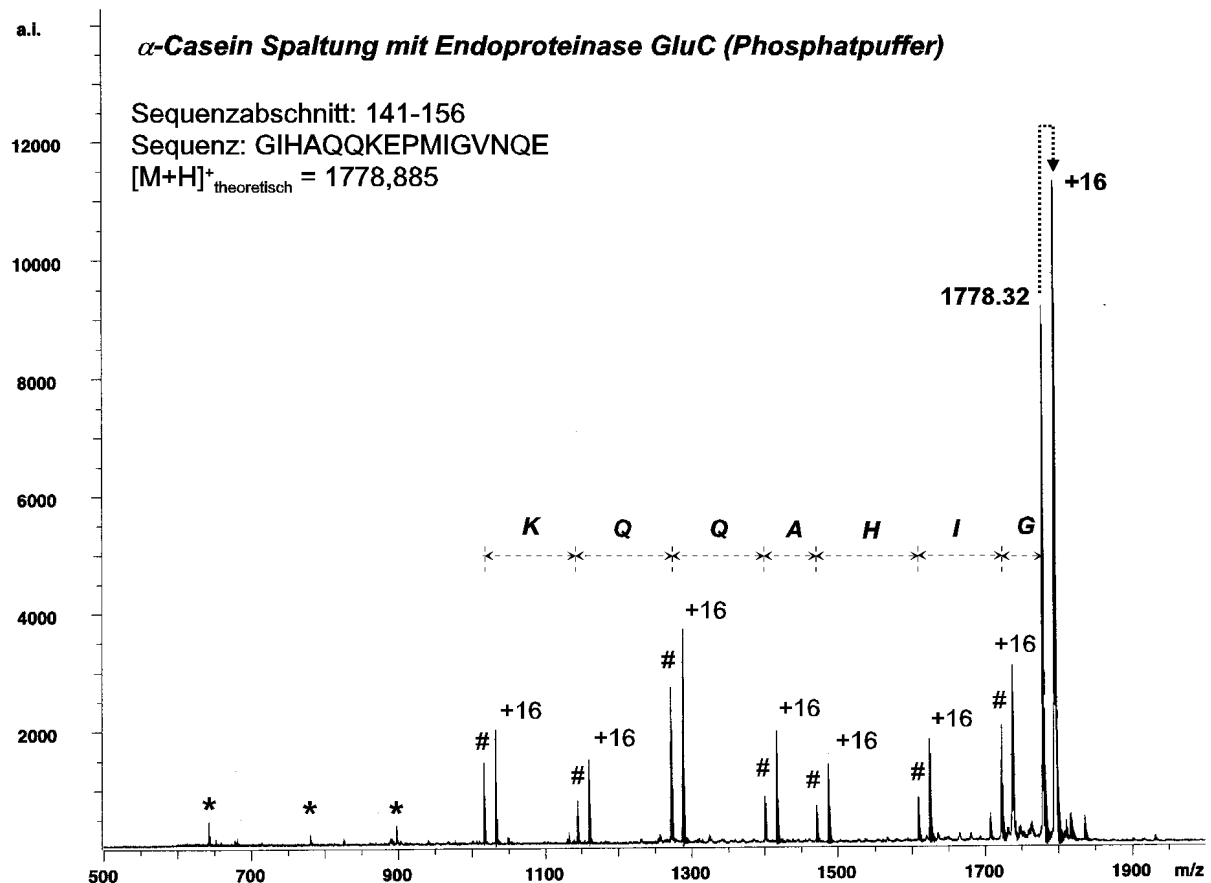
Bei einer Oxidation des Methionins beobachtet man im allgemeinen nur das Sulfoxid – die Oxidation zum Sulfon findet dagegen nicht spontan statt, kann aber über Wasserstoffperoxid erreicht werden [Schnölzer 1999]. Im Falle der Tryptophanoxidation kann die tatsächlich beobachtete Intensitätsverteilung der Reaktionsprodukte stark variieren. Als Oxidationsprodukte des Tryptophans wurden die Strukturen I und II bereits verifiziert [Simat 1994]. Aus den Spektren verschiedener, tryptophanhaltiger Peptide lässt sich mit hoher Wahrscheinlichkeit vermuten, dass der Peak mit der Massendifferenz +4 Da zum Peak des Ausgangspeptids am (offensichtlichen) Ende des Reaktionsweges, von einem Reaktionsprodukt des Acylolins stammt. Wahrscheinlich findet hier eine Decarboxylierung des Acylolins statt.

Durch die gezeigten Oxidationen können sich im Leiterspektrum charakteristische Veränderungen ergeben, die einerseits die Interpretation erleichtern, andererseits einen Hinweis auf die Anwesenheit der entsprechend oxidierten Aminosäure geben, auch wenn diese enzym-

tisch gar nicht abgespalten wird. Voraussetzung dafür ist, dass die Oxidation nur unvollständig stattgefunden hat. In diesem Fall tritt bei Methionin eine zusätzliche Peakserie mit $\Delta m = +16$ Da auf, wie zum Beispiel bei der N-terminalen Sequenzierung des α -Casein Fragments in Abbildung 64 gezeigt. Die Leiter des nicht-oxidierten Peptids ausgehend von 1778,32 Da ist mit # gekennzeichnet. Die mit * markierten Peaks können infolge der fehlenden Satellitensignale mit $\Delta m = +16$ Da von der Interpretation ausgeschlossen werden.

Die Möglichkeit einer Fehlinterpretation ist lediglich für den Fall eines vollständig oxidierten Methionins möglich. Liegt das Methionin vollständig als Methioninsulfoxid vor, so besteht an der Stelle des abgebauten Methioninsulfoxids im Leiterspektrum die Möglichkeit einer Verwechslung mit Phenylalanin, da dann nur die Massendifferenz 147 Da im Spektrum auftaucht (monoisotopische Massen: Methioninsulfoxid: 147,035 Da; Phenylalanin: 147,068 Da). Auch beim alleinigen Abbau eines Methioninsulfons (monoisotopische Masse: 163,040 Da) wäre eine Verwechslung mit dem Tyrosin (monoisotopische Masse: 163,063 Da) denkbar, allerdings ist die vollständige Oxidation zum Sulfon praktisch auszuschließen.

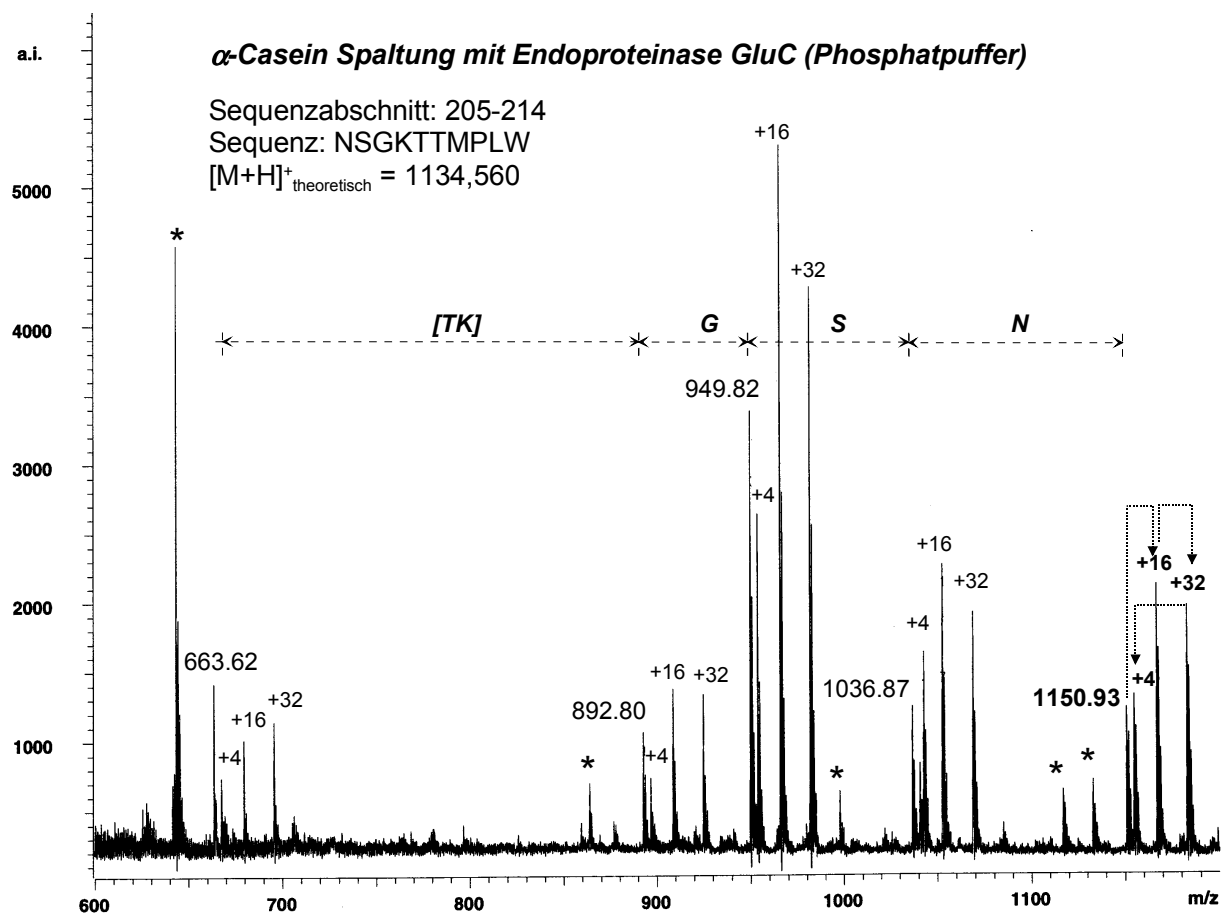
Abbildung 64: N-terminale Sequenzierung eines α -Casein Fragments mit 2,0mU APM



Bei oxidiertem Tryptophan ergibt sich durch die Vielzahl der Reaktionsprodukte ein komplexeres Peakmuster (siehe Abbildung 63). Nicht immer sind dabei wie in Abbildung 65 Signale aller möglichen Produkte zu sehen. Eine Interpretation durch Unterscheidung der Leiterpeptidsignale von anderen, nicht zur Leiter gehörigen Signalen (*) ist jedoch im Gegensatz zum oxidierten Methionin zumeist noch einfacher möglich.

Die maximalen Signalintensitäten der Leiter eines partiell oxidierten Peptids ist durch die Verteilung des Ionenstroms auf mehrere Produkte oft gegenüber der Leiter des nicht-oxidierten Peptids verringert. Der Empfindlichkeitsverlust bei der Messung wird allerdings bei ausreichender Peptidmenge für die Sequenzierung durch den diagnostischen Wert des Oxidationsmusters kompensiert. Die Massendifferenzen beim Abbau der Reaktionsprodukte des Tryptophans sind im Gegensatz zum Methionin grundsätzlich eindeutig (siehe Abbildung 63).

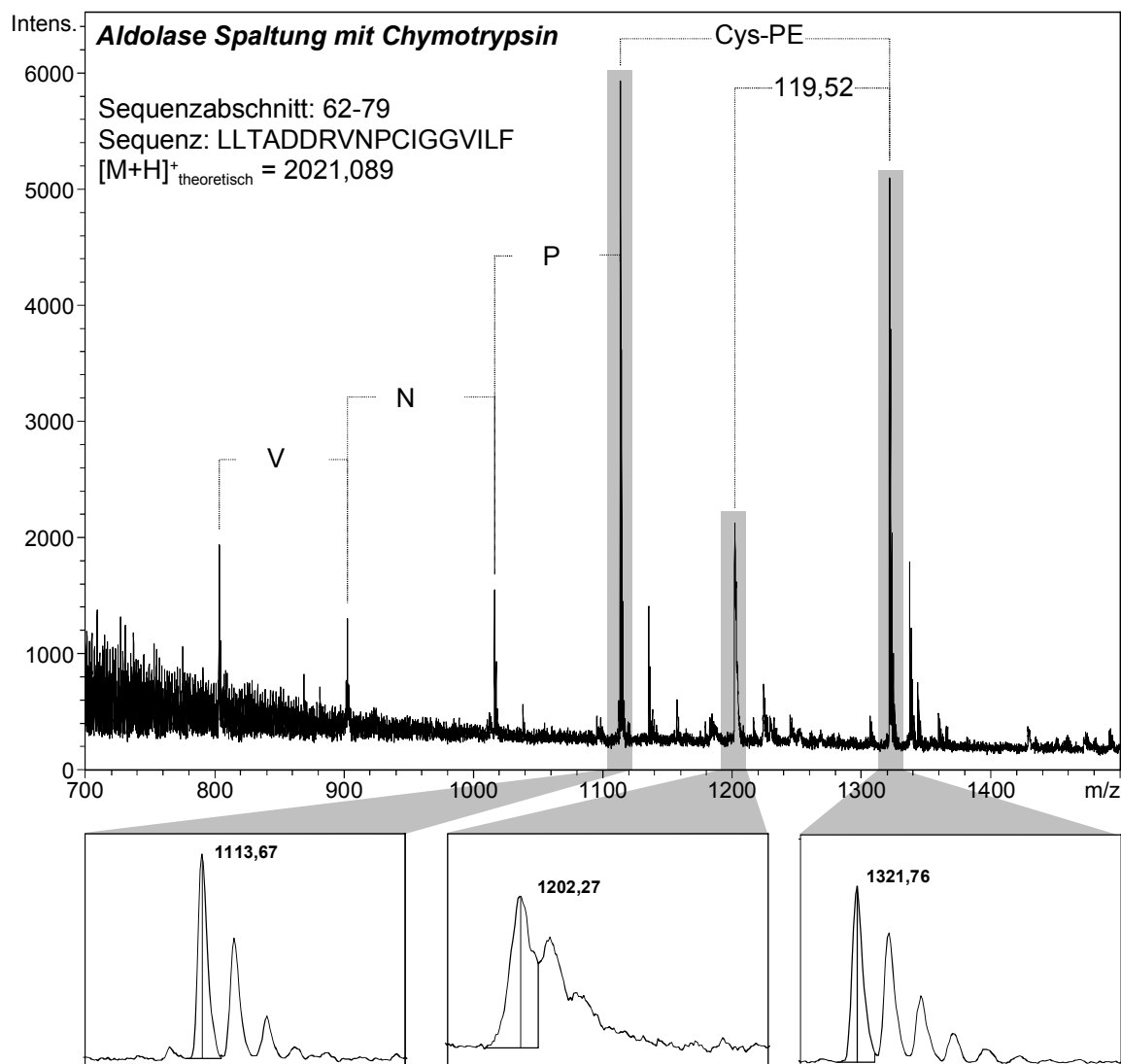
Abbildung 65: N-terminale Sequenzierung eines α -Casein Fragments mit 2,0mU APM



4.3.2 Metastabile Fragmentierungen

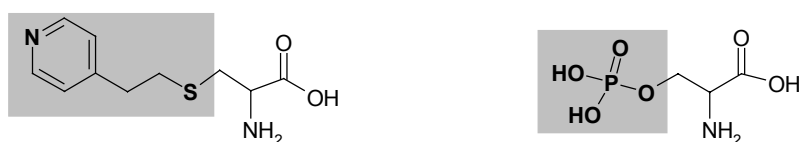
Bereits im Zusammenhang mit den Ergebnissen der post-translationalen Modifikationen (3.4 „Phosphopeptide“, S. 171) wurde erwähnt, dass charakteristische Signale metastabiler Fragmentierungen einen wichtigen Hinweis auf die betreffende Modifikation geben können. Auch die mittels Vinylpyridin alkylierten, cysteinhaltigen Peptide können als solche über eine Fragmentierung leicht identifiziert werden. Als charakteristisches Signal zeigen Leiterpeptide, die das pyridinethylierte Cystein noch enthalten, ein um ca. 119,5 Da leichteres Fragmention. Abbildung 66 zeigt dies am Beispiel der N-terminalen Sequenzierung eines Aldolasefragments (Spaltung der Aldolase mit Chymotrypsin).

Abbildung 66: Metastabile Fragmentierung bei Peptiden mit pyridinethyliertem Cystein



In den Ausschnittsvergrößerungen ist deutlich zu sehen, wie sich die Fragmentionen in ihrer Auflösung von den nicht-fragmentierten Ionen unterscheiden. Die Erkennung der Fragmentionen als solche und ihre Zuordnung zu bestimmten Leiterpeptiden ist somit im Fall des Cysteins, aber auch anderer Modifikationen, wie z.B. der Phosphorylierungen, in einfacher Weise möglich. PSD Messungen zeigen, dass die apparente Massendifferenz von 119,5 Da einer realen Massendifferenz von ca. 136,8 Da entspricht und damit für eine Abspaltung des grau unterlegten Strukturteils in Abbildung 67. Analog lässt sich bei Phosphopeptiden aus dem metastabilen Fragment mit der apparenten Massendifferenz von 87,7 Da zum Mutterion auf die Abspaltung des Phosphorsäureanteils schließen.

Abbildung 67: Strukturen von pyridinethyliertem Cystein (links) und phosphoryliertem Serin (rechts)



Bei cysteinhaltigen Peptiden ergibt sich eine zusätzliche Interpretationshilfe über die ebenfalls häufig zu beobachtende Oxidation. Wie im Falle der Methioninoxidation sieht man teilweise eine Peakserie im Abstand von $\Delta m = +16$ Da zur nicht-oxidierten Leiter.

4.3.3 Unterscheidung isobarer Aminosäuren

Beim Auftreten der Massendifferenzen 128 Da und 113 Da in den Leiterspektren ergeben sich zunächst prinzipiell Interpretationsschwierigkeiten. Obwohl die Aminosäuren Glutamin und Lysin nicht wirklich isobar sind, ist aufgrund der gegebenen Massengenauigkeit eine direkte Differenzierung nicht möglich (siehe 3.1.3 Massengenauigkeit). Während sich nach Acetylierung der ϵ -Aminogruppe des Lysins eine Möglichkeit zur eindeutigen Unterscheidung des Paares Lysin / Glutamin ergibt (Abspaltungsmasse ϵ -N-Acetyllysin ca. 170 Da), ist eine ähnliche Vorgehensweise über eine Derivatisierung zur Unterscheidung Leucin / Isoleucin nicht möglich.

Eine zusätzliche Interpretationshilfe zur Differenzierung der isobaren Aminosäuren ist jedoch in Kombination mit der Information möglich, die in der Spaltungsspezifität der angewandten Endopeptidase enthalten ist – sowohl für das Paar Lys/Gln, als auch für Leu/Ile.

Bei Spaltungen mittels Endoproteinase LysC tritt Lysin in der Regel nur direkt am C-Terminus der proteolytischen Spaltfragmente auf. Die Massendifferenz von 128 Da kann

somit bei Carboxypeptidase-Sequenzierung von Fragmenten aus Endoproteinase LysC-Spaltungen einfach zugeordnet werden. Auch Peptidsequenzen der Form $H_2N-R_n-Xcc-Xbb-Xaa-Lys-Lys-COOH$ werden häufig beobachtet, da die Abspaltung eines einzelnen Lysins zumeist nicht erfolgt. Wird in diesen Fällen CPB verwendet, so kann auch bei ausgelassenen Spaltungsstellen an zwei oder mehr aufeinanderfolgenden Lysinen eine eindeutige Zuordnung erfolgen. Innerhalb der Sequenz, also für Xbb, Xcc oder in R_n , deutet ein Peakabstand von 128 Da dagegen mit hoher Wahrscheinlichkeit auf ein Glutamin hin. Die hier ausgeführten Überlegungen für Spaltungen mit Endoproteinase LysC gelten in analoger Weise auch bei Trypsinspaltungen.

Für das isobare Paar Leu/Ile ergibt sich aus den Sequenzierungen von Proteinfragmenten einer Spaltung mit Chymotrypsin die Möglichkeit einer Unterscheidung zwischen den beiden Aminosäuren zu treffen. Das verwendete Chymotrypsin spaltet spezifisch am C-Terminus von Trp, Tyr, Phe und Leu. C-terminal sollte also die Massendifferenz 113 Da indikativ für Leucin sein, während eine innerhalb der Leitersequenz auftretende Massendifferenz von 113 Da ein Hinweis auf Isoleucin darstellt.

Bezüglich der C-terminalen Aminosäure kann eine Fehlinterpretation theoretisch nur dann auftreten, wenn Glutamin oder Isoleucin den C-Terminus des Proteins bilden. In den Versuchen zeigte keines der 98 Fragmente aus Spaltungen mit Chymotrypsin und der 243 Fragmente aus Spaltungen mit Endoproteinase LysC und Trypsin nach Vergleich mit den Datenbanksequenzen ein C-terminales Isoleucin beziehungsweise Glutamin. Die Interpretation für das terminale Leucin oder Lysin war somit in 100% der Fälle richtig.

Der Zuordnung der Massendifferenzen 113 Da und 128 Da bei deren Auftreten innerhalb der Sequenz zu Isoleucin und Glutamin ist dagegen weniger eindeutig. Vor allem bei Spaltungen mit Trypsin und Chymotrypsin findet man auch in den Proteinfragmenten häufiger ausgelassene Spaltungsstellen, so dass hier die Massendifferenzen 113 Da und 128 Da durchaus auch auf Leucin oder Lysin basieren können. Eine Auswertung der Sequenzen in 3.2.1.4 (S. 123) für Fragmente aus Chymotrypsinspaltungen zeigt, dass die eine Zuordnung der Massendifferenz 113 Da zu Ile theoretisch nur in 59% aller Fälle richtig ist. Auch bei den Fragmenten aus Trypsinspaltungen in 3.2.1.3 (S. 121) ist eine richtige Zuordnung der Massendifferenz 128 Da zu Glutamin nur in 57% aller Fälle gegeben. Bei Spaltungen mittels Endoproteinase LysC ergeben sich nach der oben geschilderten Methode allerdings sehr zuverlässige Resultate bei der Zuordnung. In 95% aller Fälle kommt die beobachtete Massendifferenz von 128 Da innerhalb der Sequenz tatsächlich durch Abspaltung eines Glutamins zustande.

4.3.4 Parallele Sequenzierungen mehrerer Peptide

Im Theorieteil wurde bereits erörtert, dass mit steigender Zahl von Fragmenten nach einer Proteinspaltung auch damit gerechnet werden muss, dass eine zunehmende Zahl der erhaltenen Fraktionen nach HPLC-Trennung mehr als nur ein Peptid enthalten (1.4.2.1, S. 56). Sofern mehrere Peptide in einer Fraktion gleichzeitig abgebaut werden können, kann es zu einer Überlappung der verschiedenen Peptidleiter im Massenspektrum kommen. Die Zuordnung der einzelnen Peaks zur jeweiligen Peptidleiter und damit das Auslesen der einzelnen Sequenzen gestaltet sich hier zwar schwieriger, ist aber dennoch möglich.

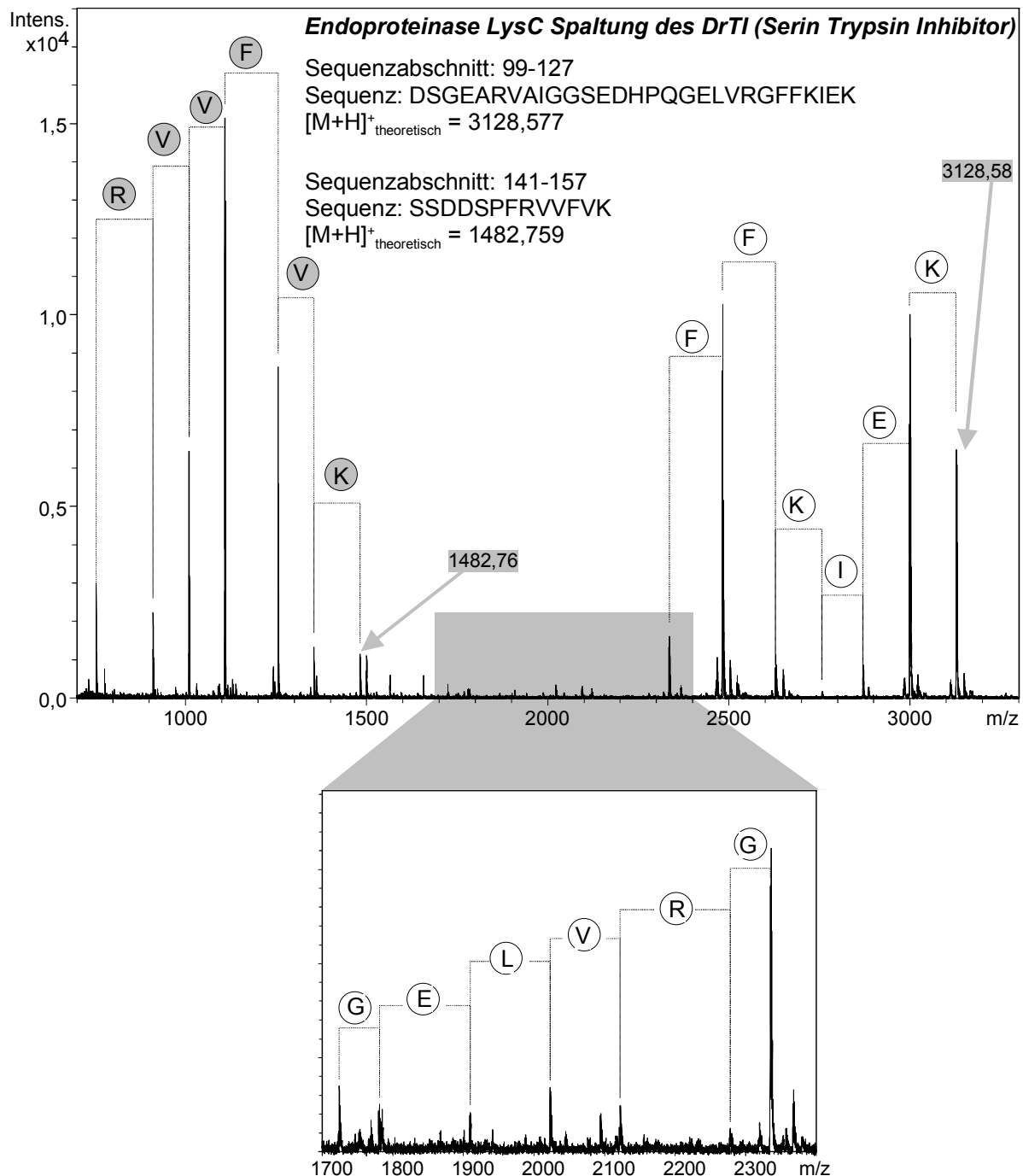
1.Fall:

Die Peptide liegen in ihren Massen relativ weit auseinander und ihre Leiterspektren interferieren somit für die Interpretation nicht. Ein Beispiel ist anhand der C-terminalen Sequenzierung zweier Peptide einer Fraktion gezeigt, die aus der Spaltung des *DrTI* (*Serin Proteinase Inhibitor*) mit Endoproteinase LysC stammt. Das gezeigte Spektrum in Abbildung 68 (folgende Seite) ist eine Überlagerung der Resultate aus der C-terminalen Sequenzierung mit 1,3 und 2,6mU CPW bei pH 4,5 in 5mM Natriumcitrat.

2.Fall:

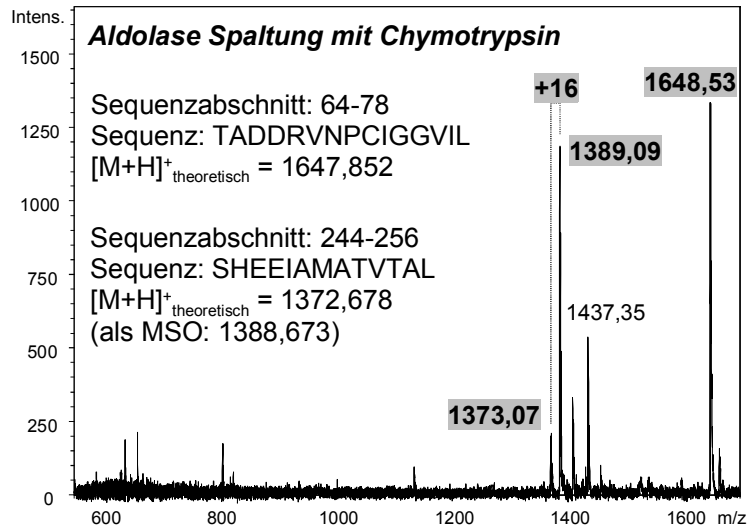
Im zweiten Fall liegen die zu sequenzierenden Peptide in ihrer Masse so nahe zusammen, dass sich ihre Leiterspektren überlagern. Man kann hier unter Umständen, wie dies zum Beispiel in Abbildung 69 (übernächste Seite) gezeigt ist, die unterschiedliche Abbaubarkeit der einzelnen Peptide mit verschiedenen Enzymen, Enzymkombinationen und Enzymkonzentrationen nutzen um die Peptidleiter weitgehend getrennt darzustellen. Die C-terminale Sequenzierung der beiden Peptide mit den Massen 1373,07 Da und 1648,53 Da erfolgte mit CPY. Bei niedrigen CPY Aktivitäten von 0,3-0,6mU beobachtet man fast ausschließlich die Sequenz des kurzen Peptids (1373,07 Da). Das Auslesen dieser Sequenz ist zudem durch eine begleitende Serie von Peaks im Abstand von +16 Da durch die Oxidation des Methionins erleichtert. Bei höheren CPY Aktivitäten von 3-6mU ist das kurze Peptid dagegen bereits weitgehend abgebaut – nur das Signal bei ca. 614 Da ist noch zu sehen. Nun dominieren die Leiterpeptide des längeren Peptids (1648,53 Da) das Massenspektrum.

Abbildung 68: Parallele, C-terminale Sequenzierung zweier Spaltpeptide des *DrTI* (*Serin Proteinase Inhibitor*) aus einer Spaltung mittels Endoproteinase LysC;

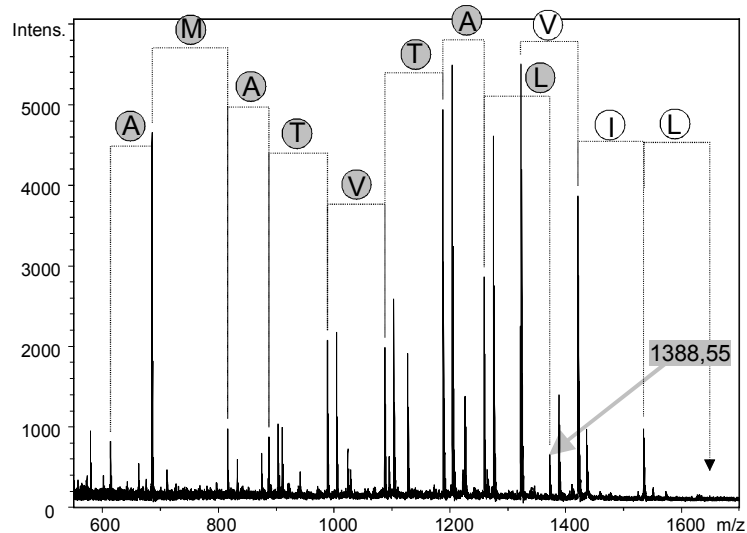


Auf diese Weise lassen sich oft auch bei überlagerten Leiterspektren durch systematische Bildung aller möglichen Massendifferenzen von Signalen höherer Masse zu Signalen niedrigerer Masse, die einzelnen Sequenzen „extrahieren“. Eine Unterstützung bei der rechnerischen Dekonvolution der einzelnen Leiterspektren durch entsprechende Software ist hier sicherlich in einfacher Weise realisierbar.

Abbildung 69: Parallele, C-terminale Sequenzierung zweier Spaltpeptide des DrTI (Serin Proteinase Inhibitor) aus einer Spaltung mittels Chymotrypsin;



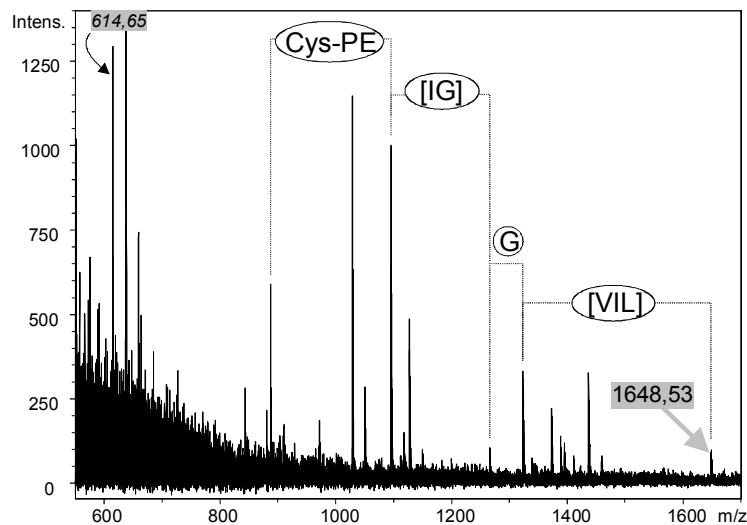
MALDI-MS der HPLC-Fraktion



Erhaltene Teilsequenz(en) bei niedriger CPY-Konzentration:

Aus Sequenzabschnitt 244-256:
AMATVTAL

Aus Sequenzabschnitt 64-78:
VIL (L -> rechnerisch durch die bekannte Ausgangsmasse des Peptids)



Erhaltene Teilsequenz(en) bei hoher CPY-Konzentration:

Aus Sequenzabschnitt 64-78:
C[IG]G

4.4 Sequenzierung modifizierter Peptide

4.4.1 Synthetische Modifikationen

Aus den Ergebnissen der unter Punkt 3.3 durchgeführten Reaktionen zeigt sich, dass Peptidderivatisierungen unter folgenden Gesichtspunkten zu einer Verbesserung einer enzymatischen Leitersequenzierungsmethode, gekoppelt mit MALDI-MS, beitragen können:

❖ Unterscheidung der Aminosäuren Lysin und Glutamin

Das unter Punkt 2.7.4 beschriebene Vorgehen zur Acetylierung gestattet eine eindeutige Unterscheidung der ansonsten nahezu isobaren Aminosäuren Lysin und Glutamin im Massenspektrum. Die Umsetzung eines Peptids mit Essigsäureanhydrid/Eisessig führt, nach den Massenspektren zu urteilen, zu einer selektiven Derivatisierung der ϵ -Aminogruppe des Lysins. Für die nachfolgende Sequenzierung stellt die Abspaltung des ϵ -N-Acetyllysins kein Hindernis dar. Die bei der Abspaltung beobachtete Massendifferenz von ca. 170 Da (monoisotopische Masse: 170,10553 Da) ist dabei charakteristisch für ϵ -N-Acetyllysine. Im Gegensatz zu anderen, bereits beschriebenen Methoden, wie z.B. von Thiede, Salnikow und Wittmann-Liebold [Thiede 1997] oder Bonetto [Bonetto 1997] ist, bei der in dieser Arbeit durchgeführten Derivatisierung, zum einen die Reaktionszeit sehr kurz (60min), zum anderen ist keine (HPLC-)Reinigung der Derivate vor der Sequenzierung erforderlich. Da bei dieser Derivatisierung auch der N-Terminus der Peptide acetyliert wird, ist im Falle einer N-terminalen Sequenzierung eine Abspaltung des blockierten N-Terminus mit Acylaminoacyl-Peptidase (EC 3.4.19.1) erforderlich.

❖ Sequenzierung kurzer Peptide und Erhöhung der Sequenzabdeckung

Eine Überlappung der MS-Signale der Peptidleitern mit den MS-Signalen von Matrixionen und Puffersalzen schränkt die maximale Sequenzinformation bei einer Leitersequenzierung zunächst prinzipiell ein. Der Exopeptidaseabbau kurzer Peptide mit ca. 4-6 Aminosäuren liefert aus dem gleichen Grund ebenfalls keine Sequenzinformation. Im Hinblick auf eine C-terminale Sequenzierung wurde dieses Problem durch eine selektive Derivatisierung des N-Terminus der zu sequenzierenden Peptide gelöst. Der durch die Derivatisierung N-terminal in das Peptid eingeführte Rest hatte eine Massenerhöhung aller Leiterpeptide bei der C-terminalen Sequenzierung zur Folge. Der bewirkte Massenshift der Leiterpeptide durch die eingeführten Gruppen lag je nach Derivatisierungsreagenz zwischen ca. 170 Da und 700 Da. Für die N-terminale Derivatisierung wurden eine Reihe unterschiedlicher Säurechloride, N-Hydroxysuccinimide (aktive Ester) und Isothiocyanate getestet. Alle Derivatisierungsreaktionen konnten in Lösung erfolgreich durchgeführt werden. Im Falle der Reaktionen mit

N-Hydroxysuccinimiden und Säurechloriden gelang auch eine direkte Derivatisierung auf dem MALDI-Target. Für eine direkte Derivatisierung auf dem MALDI-Target erscheinen aus Sicht der Leitersequenzierung allerdings nur solche Reagenzien geeignet, die nachfolgend einen problemlosen, enzymatischen Abbau des Rohprodukts erlauben (keine Störung durch überschüssiges Reagenz). Zusätzlich darf das überschüssige Reagenz bei der MALDI-MS Messung nicht stören. Diese Forderungen werden von allen Derivaten der N-Hydroxysuccinimide erfüllt. Zudem gelingen enzymatischer Abbau und nachfolgende MALDI-MS Messung bei Rohprodukten, die aus den Umsetzungen mit 6-BH-Cl, 3-BB-Cl, 4-BB-Cl und FMOC-Cl stammen. Die Peptidderivate der Säurechloride SRB-Cl, DNS-Cl und DABS-Cl sowie der Isothiocyanate erfordern eine Aufreinigung vor dem enzymatischen Abbau. Da jedoch Aufreinigungsschritte zumeist mit Verdünnung und Substanzverlusten verbunden sind, sollte nach Möglichkeit ein Reagenz gewählt werden, das bereits eine Sequenzierung des Rohprodukts erlaubt. Eine Reinigung unter Anwendung einer empfindlichen Detektion bei der HPLC (z.B. Fluoreszenz oder bei charakteristischer Absorptionswellenlänge des eingeführten Substituenten) wird daher vor allem für SRB-Peptide (große Massenverschiebung) interessant sein.

Tabelle 42 und Abbildung 47 zeigen, dass eine N-terminale Derivatisierung hinsichtlich der C-terminalen Sequenzierung fast durchweg gute bis sehr gute Ergebnisse bringt. Für die gezeigten Peptidderivate in Tabelle 42 lässt sich die C-terminale Sequenz in den meisten Fällen bis zu Di- oder Tripeptid(derivat) auslesen. Für die in Abbildung 47 gezeigten Beispiele des Bradykinins liefern alle Derivate, bis auf das α -N-Eosin-5-Bradykinin, aufgrund des eingeführten Massenshifts eine Sequenzinformation von 5 Aminosäuren (Abbau bis zum Tetrapeptidderivat). Die Sequenzinformation einer Vergleichssequenzierung des underivatisierten Bradykinins ist mit nur 2 Aminosäuren deutlich geringer. Die Isothiocyanat-Gruppe erweist sich in der Edman-Chemie unter anderem aufgrund ihrer selektiven Reaktion mit den Aminogruppen der α -Aminosäuren als besonders geeignet. Wie jedoch die Zusammenfassung der Peptidderivateigenschaften in Tabelle 43, sowie der Spektrenausschnitt in Abbildung 48 zeigen, sind die Resultate für Derivate der Isothiocyanate deutlich schlechter als für die übrigen Derivatisierungsreagenzien. Die Leiterpeptidderivate der Isothiocyanate zeigen im Massenspektrum eine extrem starke Zersetzung und die vergleichbar geringsten Signalintensitäten aller Peptidderivate. Da zudem eine HPLC-Aufreinigung für die nachfolgende Leitersequenzierung unerlässlich ist, erscheinen RB-ITC und E5-ITC trotz der großen Massenverschiebung ungeeignet für eine Derivatisierung.

Die Massendifferenzen der Zersetzungsprodukte der Isothiocyanatderivate, lassen als mögliche Reaktionswege der Zersetzung die in Abbildung 70 skizzierten Schritte plausibel erscheinen.

Abbildung 70: Mögliche Wege der Zersetzung der Isothiocyanatderivate (X = Derivatmasse in [Da])

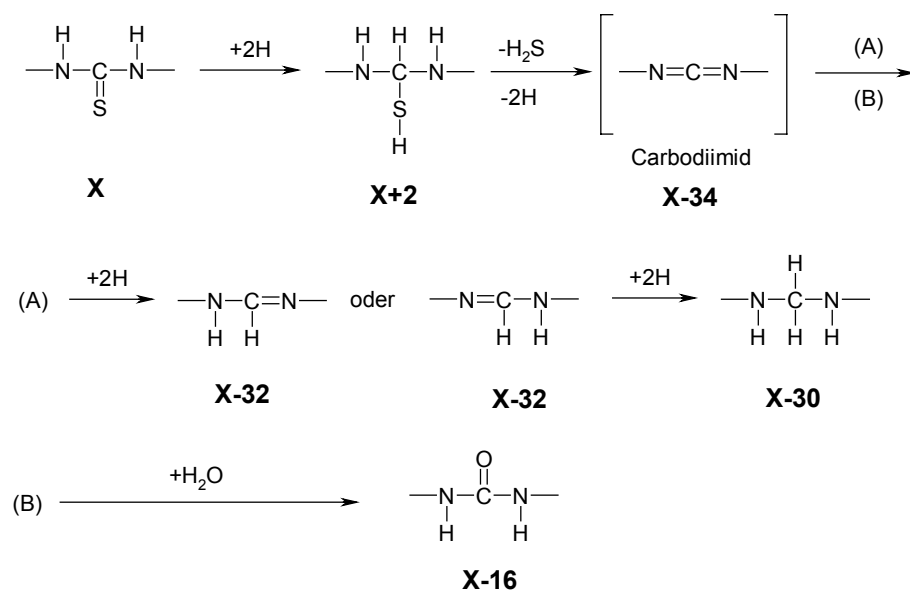


Tabelle 43 zeigt, dass auch für die Derivate der einzelnen Leiterpeptide von DABS-Cl, sowie FMOC-NHS und BBOC-NHS charakteristische Satellitenpeaks beobachtet werden können. Gegen eine bereits in Lösung stattfindende Zersetzung spricht die chromatographische Reinheit der untersuchten Peptidderivate (vor allem der Dabsylderivate): die im MS beobachteten Zerfallsprodukte eines Derivats konnten nach einer chromatographischen Trennung in keiner der untersuchten Fraktionen nachgewiesen werden. Zudem zeigt das MALDI-Massenspektrum auch nach über einem Monat unter identischen Messbedingungen ein nahezu unverändertes Spektrum: eine Zunahme der Signalintensität des Fragments auf Kosten der Intensität des eigentlichen Derivats ist nicht festzustellen. Die beobachteten Satellitensignale basieren mit großer Wahrscheinlichkeit in Analogie zum pyridinethylierte Cystein auf einer metastabilen Fragmentierung (siehe 4.3.2)

Die MS-Response der getesteten Säurechlorid- und N-Hydroxysuccinimidderivate lag im Gegensatz zu den Isothiocyanatderivaten in allen Fällen, selbst bei Messung der Rohprodukte, über der Response der jeweiligen Ausgangsprodukte. Die guten Eigenschaften der Derivate des Sulforhodamin B basieren dabei sicherlich auf dem in der Literatur bereits beschriebenen Phänomen des *charge-tagging* (siehe Abschnitt 1.4.3). Der ins Peptid eingeführte Rest enthält bereits eine fixierte, positive Ladung und erleichtert somit den Ionisierungsprozess. Für dansylierte und dabsylierte Peptide erklären sich die beobachteten, positiven Eigenschaften hinsichtlich der Massenspektrometrie ebenfalls zum Teil über die eingeführte basische Gruppe. Hier findet zwar formal nur ein Austausch zweier Aminogruppen statt, nämlich „*N-Terminus* $-NH_2$ “ gegen „*N,N-Dimethylamin*“. Allerdings wird für die primäre Aminogruppe eine tertiäre Aminogruppe eingeführt. Im Gegensatz zum Verhalten in Lösung, ist die Basizi-

tät eines tertiären Amins in der Gaspase größer, als die Basizität primärer und sekundärer Amine. Möglicherweise ist nach Ionisierung der dansylierten, dansylierten und auch sulforhodaminierten Peptide zusätzlich durch Delokalisation eine bessere Stabilisierung der positiven Ladung im Kation möglich, da alle diese Derivate ein ausgedehntes π -Elektronensystem besitzen. Aus Studien über das Fragmentierungsverhalten von Peptiden ist bekannt, dass z.B. auch bei einem Zerfall bevorzugt Ionen gebildet werden, in denen eine solche Delokalisation möglich ist [Keough 1999]. Bei den Derivaten des FMOC-NHS, BBOC-NHS und ACQ (der Chinolinstickstoff reagiert mit $pK_a=4,90$ sogar sauer) kann eigentlich nur eine solche Ladungsdelokalisierung für die gute MS-Response verantwortlich sein.

Erfolgt eine Beurteilung der Selektivität der Umsetzungen für den N-Terminus anhand der MALDI-MS Spektren der Rohprodukte, so zeigt die Gruppe der aktiven Ester (N-Hydroxysuccinimide) wiederum optimales Verhalten. Als kritisch erweist sich generell die selektive Reaktion mit einer Peptid- α -Aminogruppe bei gleichzeitiger Anwesenheit eines Lysins. Die pK_a -Werte der Peptid- α -Aminogruppe und der Lysin- ϵ -Aminogruppe liegen mit ca. 9,4 (Mittelwerte der pK_a -Werte aus α -Aminogruppe der Aminosäuren nach [Voet 1995]) beziehungsweise 10,5 sehr nahe beieinander. Insbesondere bei Umsetzungen mit FMOC-NHS und ACQ zeigen alle getesteten Peptide in hoher Ausbeute das gewünschte, N-terminal derivatisierte Peptid. Nebenreaktionen mit der ϵ -Aminogruppe des Lysins lassen sich aber auch mit N-Hydroxysuccinimiden als Derivatisierungsreagenzien nicht vollständig umgehen (vergleiche BBOC-NHS). Dagegen reagieren alle untersuchten Säurechloride in nennenswertem Umfang nicht nur mit der ϵ -Aminogruppe des Lysins, sondern auch mit den Aminosäuren Tyrosin und Histidin.

Auch bezüglich einer vereinfachten Interpretation der Leiterspektren lassen sich durch eine Peptidderivatisierung Vorteile erkennen. Die α -N-Peptidderivate der Säurechloride 3-BB-Cl, 4-BB-Cl und BBOC-NHS zeigen in den MALDI-MS-Spektren für alle Leiterpeptide die charakteristischen Isotopenmuster des Broms. Diese Leiterpeptidsignale lassen sich in den MS-Spektren sehr einfach erkennen, da sie sich von störenden, nicht zur Peptidleiter gehörenden Signalen, wie z.B. von Puffersalzen oder Matrices, deutlich unterscheiden. Bei bromhaltigen Derivaten resultieren damit auch sehr gute Empfindlichkeiten im mittleren bis unteren Femtomol-Bereich. Etwa 50-15fmol Peptidderivat lassen sich ohne Verlust an Sequenzinformation abbauen. Im allgemeinen ist die Empfindlichkeit bei der Sequenzierung der Peptidderivate etwas geringer als für Sequenzierung der entsprechend underivatisierten Peptide, insbesondere wenn eine Sequenzierung des Rohprodukts erfolgt. Resultate, wie die in Abbildung 50 bis Abbildung 52 zeigen, dass üblicherweise Peptidmengen vom unteren Pikomol- bis in den mittleren Femtomol-Bereich als Ausgangsmaterial benötigt werden.

❖ N-terminale Blockierung und Blockierung beim C-terminalen Aminosäureabbau

Im Hinblick auf eine Anwendung von Derivatisierungsreaktionen zur Verbesserung der C-terminalen Sequenzierung proteolytischer Spaltpeptide ist die Spezifität der im Protein-Fragmentierungsschritt zuvor verwendeten Endopeptidase von entscheidender Bedeutung. Da Lysin von nahezu allen getesteten Derivatisierungsreagenzien in gewissem Umfang umgesetzt wird, ist bei Proteinfragmenten aus Spaltungen mit Endoproteinase LysC oder Trypsin ein vorgeschalteter Abspaltungsschritt mit CPB erforderlich. Die Möglichkeit eines solchen Abspaltungsschritts ist allerdings positiv zu bewerten. Derivatisierte Lysine ließen sich mit den getesteten Carboxypeptidasen nur im Fall einer Acetylierung abspalten. Eine Umsetzung des Lysins mit einem der in Tabelle 39 aufgeführten Reagenzien verhinderte dagegen die Abspaltung dieses Lysins. Dabei spielt möglicherweise eine sterische Hinderung des abspaltenden Enzyms eine Rolle. Mit der vorgeschalteten Abspaltung C-terminaler Lysine in Peptiden, die aus Spaltungen mit Endoproteinase LysC oder Trypsin stammen, können folglich die „kritischen“ Aminosäuren bereits vor der Derivatisierung aus dem Peptid entfernt werden.

Der derivatisierte N-Terminus verhinderte in allen Fällen eine N-terminale Sequenzierung mittels Aminopeptidasen. Diese Beobachtung deckt sich mit den in der Literatur beschriebenen Eigenschaften der untersuchten Aminopeptidasen APM, AAP, API und LAP, wonach für deren Abbaumechanismus ein freier N-Terminus am zu sequenzierenden Peptid essentiell ist [Light 1972, Prescott 1972]. Eine N-terminale Blockierung von Proteinen hat in Organismen oft regulatorische Funktion. So soll z.B. der vorzeitige N-terminale Abbau eines Proteins vor dem Erreichen seines Bestimmungsortes durch Acetylierung oder Formylierung verhindert werden [Taylor 1993, Chen 1997]. Demzufolge ist allgemein für unspezifische Aminopeptidasen wie APM, LAP, AAP oder API der Abbau einer N-terminal modifizierten Aminosäure nicht zu erwarten.

Insbesondere im Hinblick auf die Acetylierung zur Unterscheidung von Lys und Gln besteht jedoch nach der Umsetzung die Möglichkeit, den blockierten N-Terminus mit Hilfe der Omega Peptidase Acylaminoacyl-Peptidase (EC 3.4.19.1) wieder abzuspalten. Die Peptidase spaltet dabei die N-Acetylaminosäure ab, womit anschließend eine weitere Sequenzierung durch unspezifische Aminopeptidasen wieder ermöglicht wird. Im Gegensatz dazu konnte mit keiner der eingesetzten Aminopeptidasen eine N-terminale Sequenzierung eines Derivats erreicht werden, das mit einem Reagenz aus Tabelle 39 synthetisiert wurde. Diese Derivate eignen sich also nur zur Verbesserung der C-terminalen Sequenzierung.

4.4.2 Post-translationale Modifikationen

Die Resultate aus 3.4 zeigen, dass eine Leitersequenzierung zur Identifizierung post-translationaler Modifikationen prinzipiell geeignet ist. Im Bereich der phosphorylierten Peptide war ein Abbau des Phosphotyrosins und –serins möglich (Phosphothreoninpeptide standen nicht zur Verfügung). Bei einem Vergleich des Abbaus phosphorylierter Aminosäuren bei C- und N-terminaler Sequenzierung, weist die Abspaltung der Phospho-Aminosäuren mit den getesteten Carboxypeptidasen eine starke kinetische Hemmung auf. Während sich sowohl Phosphotyrosin, als auch Phosphoserin gut mit microsomal Aminoamidase (APM) abspalten lassen, konnte C-terminal nur Phosphotyrosin mittels CPW abgebaut werden. Ein entsprechender Abbau von Phosphoserin gelang nicht. Die Probleme beim C-terminalen Abbau der Phosphopeptide zeigen eine Parallele zu den Problemen beim Abbau von Glutamin (siehe unten, Punkt 4.5 Kriterien für Sequenzabbrüche): in beiden Fällen besitzt die Seitenkette der abzuspal tenden Aminosäure unter den gegebenen Abspaltungsbedingungen um pH 5,5 eine anionische Struktur. Allerdings werden nach Tschesche [Tschesche 1977] auch die sauren Aminosäuren Asp und Glu, z.B. mit CPC (EC 3.4.16.5), gut freigesetzt. Der C-terminale Abbau des chymotryptischen Aldolase Fragments 343-357 mit der Sequenz: TPSGQAGAAASESLF beweist allerdings, dass die beobachtete Abbauehemmung auf der Phosphorylierung der entsprechenden Aminosäure basiert. Während sich aus der Sequenz der Abschnitt ASES LF abbaue lässt, erfolgt beim phosphorylierten Analogon phos-P15-1474 mit der Sequenz TPSGQAGAAASES(phos)LF lediglich der Abbau des Phenylalanins. Das Phosphotyrosin aus phos-P10-1277 YESHGKLEY(phos)A wird zwar wie gezeigt durch CPW abgebaut (Teilsequenz: EY(phos)A), allerdings lassen sich hier die nicht-phosphorylierten Analoga YESHGKLEYA und YESHGKLEY (chymotryptisches Alkoholdehydrogenase Fragment 12-20) ebenfalls wesentlich leichter und weiter sequenzieren (erhaltene Teilsequenzen: LEYA und KLEY). Die Sequenzierung des Peptids YESHGKLEYA zeigt einen möglichen Lösungsansatz für die C-terminale Sequenzierung von Phosphopeptiden. Das Peptid YESHGKLEYA wurde durch eine *on-target* Dephosphorylierung des Peptids YESHGKLEY(phos)A mit alkalischer Phosphatase erhalten. Direkt im Anschluss kann das so behandelte Phosphopeptid durch Auftragung der Carboxypeptidase (hier sb(CP-I): CPY+CPP) sequenziert werden. Durch Vergleich der Peptidmassen mit und ohne Phosphataseauftragung, sowie der Sequenzierungsergebnisse mit und ohne vorherige Phosphataseauftragung, wird man in vielen Fällen die Phosphorylierungsstelle identifizieren können. Diese Strategie ist auch für die N-terminale Sequenzierung von Phosphopeptiden denkbar.

4.5 Kriterien für Sequenzabbrüche

Betrachtet man die Sequenzierungsergebnisse der Aldolase so zeigt sich, dass 12 kurze Teilsequenzen im Protein existieren, die nach keiner der vier durchgeführten Endopeptidasespaltungen durch die Sequenzierung abgedeckt werden können. Insbesondere fällt auf, dass bei 5 dieser Teilsequenzen ein Prolin N-terminal an zweiter Stelle steht:

Teilsequenz 24-26: APG (nach Spaltung mit Endoproteinase GluC)

Teilsequenz 91-93: RPF (nach Spaltung mit Endoproteinase GluC, Trypsin)

Teilsequenz 191-193: LPD (nach Spaltung mit Endoproteinase GluC)

Teilsequenz 234-236: TPG (nach Spaltung mit Endoproteinase LysC)

Teilsequenz 261-262: PP (nach Spaltung mit Chymotrypsin)

Eine genauere Untersuchung der Sequenzresultate aus 3.2.1 zeigt, dass hier bei allen Peptiden, aus denen diese Tripeptidsequenzen stammen, die N-terminale Sequenzierung der Aminosäure vor dem Prolin, also die Spaltung der Iminbindung, scheitert und somit die Sequenzierung abbricht. Insgesamt erfolgt bei ca. 300 N-terminal sequenzierten Peptiden in 14% aller Fälle ein Sequenzabbruch dort, wo die abzuspaltende Aminosäure über eine Iminbindung mit einem Prolin verknüpft ist. Nach [Doolittle 1989] (siehe unten Tabelle 54) beträgt der Anteil von Prolin in Proteinen im Mittel 5,2%. Die Berechnung des Prolinanteils eines hypothetischen Proteins, gebildet aus allen unter 3.2.1 aufgelisteten Peptiden ergibt einen Wert von 5,7%. Damit stimmt der mittlere Prolingehalt der sequenzierten Peptide sehr gut mit dem theoretischen Wert überein. Aus statistischer Sicht sollte daher die Häufigkeit eines N-terminalen Sequenzabbruchs am Prolin in Position Xbb der Sequenz $H_2N-Xaa-Xbb-Xcc-R_n-COOH$ ebenfalls um 5-6% liegen. Tatsächlich findet jedoch bei den durchgeführten Experimenten ein Sequenzabbruch fast dreimal so häufig statt. Dies zeigt, dass die Iminbindung bei der Sequenzierung mit den getesteten Peptidasen in der Tat ein besonderes Problem darstellt.

Es stellt sich die Frage, ob sich für die enzymatische Leitersequenzierung, ebenso wie im Falle des Prolin und der Iminbindung, weitere einzelne Aminosäuren oder Aminosäuremotive in einer Sequenz als kritisch hinsichtlich des Abbaus erweisen. In einer Sequenzabfolge $H_2N-Xaa-Xbb-Xcc-R_n-COOH$ bei N-terminaler Sequenzierung beziehungsweise $H_2N-R_n-Xcc-Xbb-Xaa-COOH$ bei C-terminaler Sequenzierung mit Monopeptidylpeptidasen wird dabei sicherlich in den meisten Fällen der Aminosäure Xaa die größte Bedeutung zukommen. Diese

These wird bei einem Blick auf die in Tabelle 2 und Tabelle 3 (S. 16ff.) zusammengestellten, bisher bekannten Exopeptidaseeigenschaften weitgehend gestützt. Da allerdings auch sekundäre Wechselwirkungen der Substraterkennung einen Einfluss auf das beobachtete Abspaltungsverhalten haben können (siehe auch 1.1.3), sind die exakten Gründe für einen Sequenzabbruch möglicherweise wesentlich komplexer. So wird auch in Tabelle 2 und Tabelle 3 in Einzelfällen auf mögliche Einflüsse von Xbb auf die Abspaltung verwiesen. Durch [Sprössler 1971] wurde z.B. gezeigt, dass bei der C-terminalen Sequenzierung mit Carboxypeptidase C bei gleichem Xaa, aber unterschiedlichem Xbb die relativen Abspaltungsraten zum Teil von 100% bis 6% variieren können (100% entspricht dem Xbb mit der höchsten Abspaltungsrate). Besonders Glycin in Position Xbb erschwert den Abbau von Xaa. Das Peptid der Sequenzabfolge $H_2N-R_n-Xcc-Gly-Pro-COOH$ konnte überhaupt nicht weiter abgebaut werden. Auch für Glycin selbst in Position Xaa wurden durch die Autoren nur sehr geringe Abspaltungsraten beobachtet: im Fall des Peptids $H_2N-R_n-Xcc-Pro-Gly-COOH$ minimal 1% und für $H_2N-R_n-Xcc-Leu-Gly-COOH$ maximal 8%. Auch wenn diese Effekte und sekundären Wechselwirkungen hier nicht weiter berücksichtigt werden können, sind die folgenden Daten in der Praxis und für eine zukünftige Verbesserung der Methode ein wichtiger Anhaltspunkt. Tabelle 54 lässt sich wie folgt interpretieren: Die ersten beiden Spalten stellen die experimentell ermittelte Häufigkeit jeder Aminosäure aus den proteolytischen Spaltfragmenten (S. 114ff.) dem entsprechend theoretischen Wert nach [Doolittle 1989] gegenüber. Für den berechneten experimentellen Wert wurde auf ein hypothetisches Protein, bestehend aus allen untersuchten und der Sequenzierung unterworfenen Peptide (siehe 3.2.1), bezogen. Dieses hypothetische Protein setzt sich aus 365 Peptiden zusammen und besteht somit aus 5131 Aminosäuren. Theoretisch nach [Doolittle 1989] wurde die Häufigkeit des Auftretens zwar auf eine wesentlich größere Grundgesamtheit bezogen (300688 Aminosäuren), jedoch zeigt ein Vergleich, dass die experimentellen Häufigkeiten trotzdem gut mit den theoretischen Werten übereinstimmen. Die durchschnittliche Abweichung Δ (dritte Spalte) von theoretischer zu experimenteller Häufigkeit liegt bei nur 0,75%.

Die letzten vier Spalten geben die Häufigkeit des Abbaus und des Sequenzabbruchs für eine bestimmte Aminosäure wieder – jeweils getrennt nach N- und C-terminaler Sequenzierung. Die **Häufigkeit des Sequenzabbruchs** gibt an, wie oft die betreffende Aminosäure in Position Xaa im Falle des Sequenzabbruchs vorzufinden ist. Es wurde bei den Bezugsgrößen die Tatsache berücksichtigt, dass für die Fragmente der Alkoholdehydrogenase und des Cytochrom C nach Proteinspaltungen den Endoproteinasen LysC und Chymotrypsin keine N-

terminalen Sequenzierungen durchgeführt wurden. Dementsprechend wurde C-terminal auf alle 365 Fragmente (1176 abgebaute Aminosäuren) bezogen, N-terminal aber nur auf 303 Fragmente (655 abgebaute Aminosäuren). Die **Abbauhäufigkeit** gibt Auskunft darüber, wie oft die betreffende Aminosäure in den abgebauten Teilsequenzen auftritt.

Tabelle 54: Häufigkeit des Vorkommens bestimmter Aminosäuren in den sequenzierten Peptiden, sowie N- und C-terminale Abbau- und Sequenzabbruchshäufigkeit an bestimmten Aminosäuren

Aminosäure	Häufigkeit [%]		Δ^2 [%]	N-terminale Häufigkeit [%]		C-terminale Häufigkeit [%]	
	über alle sequenzierten Peptide	theoretisch ¹		Abbau	Sequenzabbruch	Abbau	Sequenzabbruch
Gly (G)	7,2	8,1	0,9	7,0	7,3	7,3	8,5
Ala (A)	7,8	8,9	1,1	11,8	8,6	7,2	2,2
Val (V)	6,6	6,4	0,2	5,2	5,9	7,5	0,8
Leu (L)	9,1	8,9	0,2	10,2	5,9	10,4	4,7
Ile (I)	5,3	5,4	0,1	5,2	5,6	5,4	2,2
Met (M)	2,2	1,3	0,9	0,9	0,7	1,2	0,8
Pro (P)	5,2	5,7	0,5	1,8	4,3	3,6	3,8
Phe (F)	3,9	3,3	0,6	2,4	1,3	4,5	2,7
Trp (W)	1,4	0,6	0,8	0,5	0,7	0,5	0,8
Ser (S)	6,8	5,8	1,0	9,0	4,6	4,3	2,2
Thr (T)	5,9	5,1	0,8	5,5	6,3	3,8	2,7
Asn (N)	4,3	2,9	1,4	2,7	3,6	3,0	1,9
Gln (Q)	4,3	4,5	0,2	6,0	1,0	4,8	2,7
Tyr (Y)	3,2	3,5	0,3	4,3	5,3	3,7	5,2
Cys (C)	1,9	2,3	0,4	3,2	2,6	2,5	3,3
Lys (K)	5,9	7,3	1,4	5,8	6,3	10,5	21,1
Arg (R)	5,1	3,8	1,3	3,1	7,9	4,2	8,8
His (H)	2,3	2,8	0,5	2,1	6,3	1,9	2,7
Asp (D)	5,3	5,5	0,2	6,6	5,9	4,3	4,7
Glu (E)	6,3	8,4	2,1	6,7	8,6	9,5	15,6

¹ Theoretische Häufigkeit des Vorkommens nach [Doolittle 1989], berechnet aus einer nicht-redundanten Datenbank mit 300688 Aminosäuren.

² Δ : Abweichung von theoretischer zu experimenteller Häufigkeit des Auftretens

Die erhaltenen Werte für die Abbauhäufigkeit und die Sequenzabbruchhäufigkeit sind nun unter folgenden Gesichtspunkten zu betrachten: Erfolgt der Sequenzabbruch rein statistisch, so sollten sowohl die Abbauhäufigkeit, als auch die Abbruchhäufigkeit für alle Aminosäuren

einen der Häufigkeit ihres Vorkommens vergleichbaren Wert besitzen. Ein stark erhöhter Wert für die Sequenzabbruchhäufigkeit einer bestimmten Aminosäure in Position Xaa deutet darauf hin, dass die verwendeten Peptidasen oder Peptidasekombinationen Schwierigkeiten beim Abbau der betreffenden Aminosäure haben. Diese Hypothese gilt jedoch nur dann als gut abgesichert, wenn im Gegenzug auch die Abbauhäufigkeit dieser Aminosäure gering ist. Für die Beurteilung einer Erhöhung oder Erniedrigung der beobachteten Häufigkeit ist zudem zu berücksichtigen, dass auf eine wesentlich kleinere Zahl an Aminosäuren bezogen wird. Bei der Abbruchhäufigkeit entspricht die Zahl der betrachteten Aminosäuren der Zahl der sequenzierten Fragmente, also 365 Aminosäuren C-terminal und 303 Aminosäuren N-terminal. Bei der Abbauhäufigkeit sind es immerhin 1176 beziehungsweise 655 Aminosäuren. Speziell bei der Abbruchhäufigkeit wird also auf eine wesentlich kleinere Aminosäurezahl bezogen und es sind daher grundsätzlich etwas größere Abweichungen von den erwarteten Häufigkeitswerten in den Spalten zwei und drei von Tabelle 54 zu erwarten. Unter Berücksichtigung dieser Tatsache erweisen sich eigentlich nur die grau unterlegten Werte in Tabelle 54 bezüglich ihrer Abweichungen als statistisch signifikant.

Einen sehr hohen Wert für die N-terminale Abbruchhäufigkeit, bei normaler Abbauhäufigkeit besitzt lediglich das Histidin. Alle anderen Aminosäuren zeigen innerhalb einer tolerierbaren Fehlergrenze eine Abbruchhäufigkeit, die dem durchschnittlich erwarteten Wert für die Summe aller sequenzierten Peptide entspricht. Ganz im Gegenteil fallen sogar mit Glutamin und Leucin zwei Aminosäuren mit besonders niedriger Abbruchhäufigkeit auf. Für Lysin, Arginin und Glutamin ist das Bild durch die Spaltungen mit Endoproteinase LysC und Trypsin, sowie Endoproteinase GluC etwas verzerrt. Die genannten Aminosäuren befinden sich (Endopeptidasen-bedingt) überproportional oft am C-Terminus der sequenzierten Spaltfragmente. Damit sollte sowohl ihre N-terminale Abbauhäufigkeit, als auch die N-terminale Abbruchhäufigkeit erniedrigt sein. Tatsächlich liegen die Werte jedoch scheinbar normal und im Falle der Abbruchhäufigkeiten von Arg und Glu damit sogar deutlich zu hoch. Da der N-terminale Abbau vor Prolin, wie bereits im vorangegangenen Abschnitt gezeigt, meist dann schon stehen kommt, wenn Prolin in Xbb Position ist (d.h. Pro in Position Xaa kommt selten vor), ist die geringe Abbauhäufigkeit des Prolins gut verstehen (Abbruchhäufigkeit hier ca. 14% !). Allerdings findet auch in 1,8% aller Fälle ein Abbau des Prolins statt und 4,3% aller Fälle kommt es erst dann zum Sequenzabbruch, wenn Prolin in Xaa Position ist. Ein Teil der Abbauhäufigkeit ist dabei allerdings auf Spaltpeptide zurückzuführen, die das Prolin bereits N-terminal enthalten, z.B. im Aldolase Fragment 230-242: PNMVTPGHACTQK (abgebauter Teil ist unterstrichen). Der Abbau eines solchen N-terminalen Prolins stellt für APM,

AAP und API kein Problem dar, da zuvor keine Iminbindung gespalten werden muss. Aber auch innerhalb einer Sequenz konnten die Iminbindung oder das Prolin in Einzelfällen deutlich abgebaut werden, so z.B. im β -Casein Fragment 185-191 (VLPVPQK) oder im β -Lactoglobulin Fragment 50-61 (AQSAPLRVYVEE). Der Abbau gelang in solchen Fällen jedoch nur mittels APM. Die Iminbindung wirkt damit aber nicht grundsätzlich blockierend für den Abbau.

Die gegebene Problematik durch den erschwerten Abbau der Xaa-Pro-Bindung könnte durch die Verwendung einer spezifischen Aminopeptidase umgangen werden. X-Pro Aminopeptidase (Prolin Aminopeptidase; EC 3.4.11.9; Tabelle 3, S. 17) spaltet spezifisch die Iminbindung. Diese Peptidase ist jedoch kommerziell nicht verfügbar. Die einzige käufliche Imino-peptidase ist Prolidase (Imidodipeptidase oder Prolindipeptidase; EC 3.4.13.9). Diese Imino-peptidase spaltet jedoch nur die Iminbindung in Dipeptiden.

Auf der C-terminalen Seite fallen zunächst vor allem die extrem hohen Werte für Lysin und Glutaminsäure, sowie in etwas geringerem Maße für Arginin auf (schwarz unterlegte Felder). Bei der Beurteilung dieser Werte sind jedoch wiederum die Spaltspezifitäten der gewählten Endopeptidasen zu berücksichtigen. Insgesamt wurden aus Spaltungen mit Endoproteinase LysC und Trypsin 144 Fragmente mit C-terminalem Lysin erhalten. Die aus Spaltungen mit Endoproteinase GluC (Phosphatpuffer) stammenden Fragmente besitzen in 65 Fällen Glutaminsäure am C-Terminus. Somit tragen über die Hälfte aller C-terminal sequenzierten Fragmente am C-Terminus ein Lysin oder Glutaminsäure. Die C-terminale Häufigkeit der Asparaginsäure entspricht dagegen im ihrem Wert weitgehend der theoretisch zu erwartenden Häufigkeit dieser Aminosäure in Proteinen – trotz Spaltung mit Endoproteinase GluC (Phosphatpuffer). Durch die erhöhten Werte für Lysin oder Glutaminsäure werden in der Folge die Anteile eines Sequenzabbruchs an diesen beiden Aminosäuren gegenüber den anderen Aminosäuren stark erhöht. Die Erhöhung der Werte für Lysin und Glutaminsäure geht auch zu Lasten der übrigen Aminosäuren, deren um 2-3% erniedrigte Werte für Sequenzabbruch- und Abbauhäufigkeit gegenüber dem durchschnittlichen Vorkommen somit als normal anzusehen sind. Betrachtet man die Abbauhäufigkeit von Lysin ohne die Fragmente aus Spaltungen mit Endoproteinase LysC beziehungsweise ohne die Fragmente aus Spaltungen mit Trypsin, welche C-terminal ein Lysin tragen, erhält man folgende Werte:

Lys (Abbauhäufigkeit): 5,8% (bezogen auf 718 AS)

Lys (Sequenzabbruchhäufigkeit): 6,6% (bezogen auf 221 Fragmente)

Diese Werte zeigen, dass das Abbauverhalten der Carboxypeptidasen bezüglich des Lysin grundsätzlich als normal zu bezeichnen ist. Andererseits haben Sequenzierungen mit CPB oder Peptidasekombinationen mit CPB gezeigt, dass alle Lysin- oder Argininreste mit dieser Carboxypeptidase abgebaut werden können. In der Praxis stellt der Abbau von Lysin oder Arginin somit ohnehin kein Problem dar. Für den (hier nicht zutreffenden) Fall, dass CPB auf alle zu sequenzierenden Peptide angewendet wird, sind daher für die Sequenzabbruchhäufigkeiten an Lysin und Arginin extrem niedrige Anteile zu erwarten. Auf der anderen Seite sollten dann die zu erwartende Abbauhäufigkeit – insbesondere bei Spaltungen mit Endoproteinase LysC und Trypsin – gegenüber anderen Aminosäuren signifikant erhöht sein. Analog ist es auch für Glutaminsäure notwendig, eine genauere Untersuchung der Werte unter Ausschluss von Fragmenten aus Spaltungen mit Endoproteinase GluC vorzunehmen. Hierbei ergibt sich dann folgendes Bild:

Glu (Abbauhäufigkeit): 7,5% (bezogen auf 883 AS)

Glu (Sequenzabbruchhäufigkeit): 10% (bezogen auf 289 Fragmente)

Der Wert für die Abbauhäufigkeit erscheint normal. Signifikant erhöht ist jedoch die Häufigkeit eines Sequenzabbruchs (C-terminal) vor Glutaminsäure. Im Gegensatz zur Spaltung der Iminbindung zum Prolin, ist eine Lösung durch Anwendung einer spezifischen Carboxypeptidase hier nicht möglich. Tabelle 2 (1.1.1, S. 16) zeigt, dass speziell für die C-terminale Abspaltung der Aminosäuren mit sauren Seitenketten, wie Glutaminsäure oder Asparaginsäure, keine spezifische Carboxypeptidase bekannt ist. Carboxypeptidase G (EC 3.4.17.11) spaltet spezifisch nur dann, wenn eine γ -Glutamylbindung vorliegt !

Im Gegensatz zu den meisten anderen, nur leicht erniedrigten Werten (aufgrund der hohen Werte von Lys und Glu) sind Alanin, Serin, Leucin und vor allem Valin signifikant selten für einen Sequenzabbruch verantwortlich. Diese Beobachtung lässt den Rückschluss zu, dass sich diese Aminosäuren (vor allem die unpolar aliphatischen) vergleichsweise gut abbauen lassen. Bei Valin und Leucin liegen die Abbauhäufigkeiten sogar leicht über den Werten für die Häufigkeit des Vorkommens, was aufgrund der hohen Werte von Lys und Glu bemerkenswert ist. Die Häufigkeiten eines Sequenzabbruchs vor Glycin und Tyrosin erscheinen dagegen gerade aus diesem Grund signifikant erhöht. Cystein zeichnet sich zwar ebenfalls häufiger für einen Sequenzabbruch verantwortlich als zu erwarten, jedoch liegt für diese Aminosäure auch die Abbauhäufigkeit über dem durchschnittlichen Vorkommen.

Insgesamt ist es nur schwer möglich über die Anwendung einer nicht quantitativ arbeitenden Messmethode wie der MALDI-Massenspektrometrie bestimmte Aminosäuren für Sequenzabbrüche verantwortlich zu machen. Hinzu kommt, dass in meisten Fällen der Abbau solcher problembehafteter Aminosäuren eigentlich „nur wesentlich langsamer“ abläuft als der Abbau anderer Aminosäuren. Eine absolute Hemmung wird seltener gegeben sein. Da die Methode der MALDI-MS für viele Peptide auch bei geringen Mengen sehr empfindlich ist, lässt sich die Leitersequenz selbst bei Abbauproblemen an definierten Sequenzpositionen oft noch eine oder gar mehrere Aminosäuren über die betreffende Stelle hinaus auslesen. Diese, durchaus als Vorteil anzusehende Eigenschaft der Leitersequenzierungsmethode erschwert jedoch die Interpretation der Sequenzdaten unter 3.2.1 im Hinblick auf eine Identifizierung der für Abbauschwierigkeiten verantwortlichen Sequenzpositionen oder –abschnitte.

Bei Cystein ist für alle Peptide zu berücksichtigen, dass diese Aminosäure in Form eines S-pyridinethylierten Derivats vorkommt. Dieses Derivat lässt sich sehr gut enzymatisch – vor allem N-terminal durch die Exopeptidase APM abbauen.

4.6 Sequenzlängen und Anwendungsbereich

Über die Verteilung der erhaltenen Sequenzlängen und Sequenzabdeckungen lässt sich auch das potentielle Anwendungsgebiet enzymatischer Leitersequenzierungen definieren.

Ist die Sequenz des Proteins bereits als Eintrag in einer Proteindatenbank wie SwissProt oder PIR oder einer Nucleotiddatenbank wie EMBL oder GenBank vorhanden, so ist eine Sequenzlänge von 4-6 Aminosäuren für eine erfolgreiche Datenbanksuche zumeist ausreichend. Tabelle 55 zeigt dies für die Datenbanksuche mit einigen Teilsequenzen, die aus Proteinen stammen die mittels Endoproteinase LysC gespaltenen wurden. Die Datenbanksuche wurde über das Programm „MS-Pattern“ durchgeführt. Dieses stammt aus dem Programmpaket „Protein Prospector“ der UCSF Mass Spectrometry Facility und ist via World Wide Web unter der Adresse <http://prospector.ucsf.edu/ucsfhtml3.4/mspattern> frei zugänglich. Folgende Datenbanken wurden bei der Suche verwendet:

- ❖ SwissProt.12.06.2000 (90939 Einträge)
- ❖ Genpept.12.06.2000 (635529 Einträge)
- ❖ pdbEST.others.12.06.2000 (2021600 Einträge)

Dabei wurde als zusätzliche nur Information der Organismus spezifiziert, aus dem das gesuchte Protein ursprünglich isoliert wurde. Eine genauere Einschränkung der Proteinmasse erfolgt nicht, trotz der Tatsache, dass diese Größe in der Regel ebenfalls bekannt ist. Der

eingetragene Massenbereich zwischen 1 und 100 kDa für die Datenbanksuche in SwissProt und Genbank schließt ebenfalls gerade einmal 4-6% aller Sequenzen in den durchsuchten Datenbanken aus. Der pI-Bereich des gesuchten Proteins wurde nicht eingeschränkt. Der Tatsache, dass zwischen Leucin und Isoleucin bei den Spaltungen mit Endoproteinase LysC auf keinen Fall unterschieden werden kann, wird durch einen entsprechenden Eintrag bei der Suche Rechnung getragen (mehrdeutiges Suchmuster: wenn in der gesuchten Sequenz ein Leu steht, werden auch Sequenzen mit Ile in dieser Position gesucht und umgekehrt). Zudem wurden für jede Suche 1-2 Aminosäure-Fehlzuordnungen (Eintrag „mismatched Amino Acids“) zugelassen.

Tabelle 55: Resultate der Datenbanksuche mit Teilsequenzen aus den LysC-Fragmenten

gesuchtes Protein	Teilsequenz maximal ¹	Bereich ²	AS		SwissProt minimal ³		Genbank minimal ³
Cytochrom C P00004	TYTDANK	C	7	+	DANK	+	YTDANK
	EY(L/I)ENPK	C	7	+	ENPK	+	EY(L/I)EN
Myoglobin P02188	Y(L/I)EF(I/L)SD	N	7	+	Y(L/I)EF(I/L)S	+	Y(L/I)EF(I/L)SD
	V(L/I)NVWGK	C	7	+	VWGK	+	(L/I)NVWGK
β-Lactoglobulin P02754	VDDEA(L/I)EK	C	8	+	DEA(L/I)EK	+	DEA(L/I)EK
	Y(L/I)(L/I)FCME	N	8	+	Y(L/I)(L/I)FC	+	Y(L/I)(L/I)FCM
β-Casein P02666	FQSEEQQQ	N	8	+	FQSEE	+	FQSEE
	EDE(L/I)QDK	C	7	+	DELQDK	+	DELQDK
	VRGPFP(I/L)	I	7	+	RGPFP(I/L)	+	RGPFP(I/L)
Aldolase P00833	AAQEEYVK	G	8	+	AQEEYVK	+	AQEEYVK
	RA(L/I)QASA(L/I)K	C	9	+	ASA(L/I)K	+	ASA(L/I)K
	CAQY(K/Q)K	C	6	+	AQY(K/Q)K	0	CAQY(K/Q)K
	DDRVNPC	I	7	+	VNPC	+	RVNPC
Rinderserum Albumin P02769	SE(I/L)AHRFK	G	8	+	SE(I/L)AH	+	SE(I/L)AH
	TVMENF	N	6	+	TVME	+	TVMEN
	VAFVDK	C	6	+	AFVDK		VAFVDK
	Y(L/I)YE(I/L)ARR	N	8	+	Y(L/I)YE	+	Y(L/I)YE(I/L)A
	VAS(L/I)RETYG	N	9	+	VAS(L/I)RE	+	VAS(L/I)RE
	TCVADES	N	7	+	TCVAD	+	TCVAD

+: Zuordnung eindeutig;

0: Zuordnung möglich, aber nicht eindeutig;

¹ Datenbanksuche mit der maximalen Teilsequenz des Proteins wie sie aus einer N- oder C-terminalen Sequenzierung des Spaltpeptids erhalten wurde

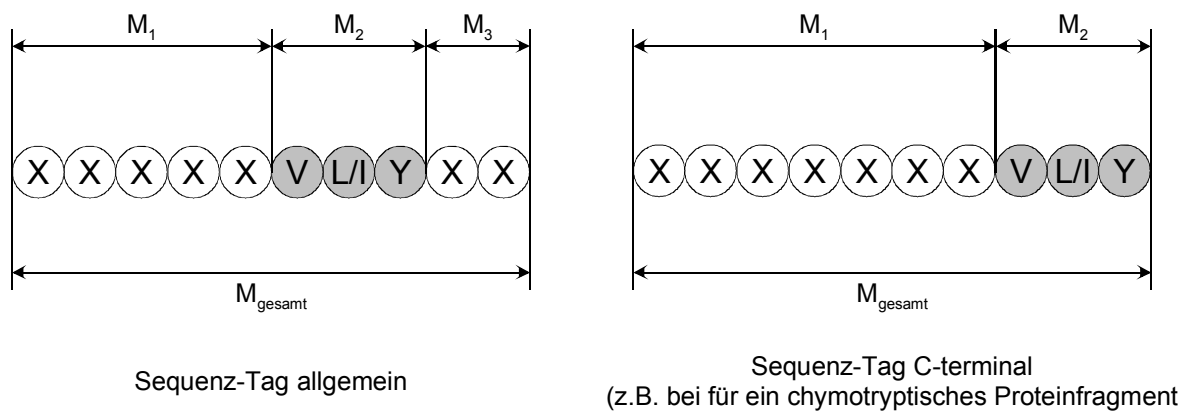
² Bereich der Sequenz aus dem ursprünglich sequenzierten Proteinfragment; C: C-terminaler Sequenzabschnitt; N: N-terminaler Sequenzabschnitt; I: interner Sequenzabschnitt; G: komplette Sequenz eines Spaltpeptids

³ Die Datenbanksuche in SwissProt/GenPept mit dieser verkürzten Teilsequenz liefert ebenfalls noch das gleiche Resultat wie die Datenbanksuche über die maximale Teilsequenz

Die obige Tabelle zeigt, dass bei Datenbanksuche über die nicht-redundante Datenbank SwissProt bereits ein sehr kurzer Sequenzabschnitt von nur 4-5 Aminosäuren das gesuchte Protein mit der höchsten Wahrscheinlichkeit zurückgibt.

Alle Teilsequenzen beinhalten außerdem zusätzliche Informationen für eine Datenbanksuche über die Peptidmasse. Das Prinzip, die Information über die Masse mit in die Suche einzubeziehen wird als sogenannte „*Tag-Sequenzierung*“ bezeichnet. Dabei werden zusätzlich auch die in Abbildung 71 gezeigten Masseninformationen M_{gesamt} , M_1 , M_2 und M_3 aus der Sequenzierung für die Datenbanksuche verwendet. Diese Teilmassen fallen bei der Sequenzierung über eine massenspektrometrische Methode wie der MALDI-MS automatisch an.

Abbildung 71: Informationen eines massenspektrometrisch erzeugten Sequenztags



Mit den zusätzlichen Masseninformationen genügt häufig bereits ein extrem kurzer Sequenzabschnitt von 2-3 Aminosäuren für eine erfolgreiche Identifizierung des Proteins in der Datenbank [Mann 1994] auch wenn, wie im Falle des in Abbildung 71 gezeigten terminalen Sequenz-Tags, die Information auf drei Teilmassen M_{gesamt} , M_1 , M_2 reduziert ist. Auch für die Suche über GenPept und EST Datenbanken sind diese Sequenzinformationen ausreichend. Wird allerdings ausschließlich mittels Sequenzinformation gesucht, so zeigt sich bei Suchen in EST Datenbanken erst mit etwas längeren Sequenzabschnitten von 5-6 Aminosäuren eine gute Trefferwahrscheinlichkeit für das gesuchte Protein.

Dennoch ist in der Summe für alle sequenzierten Proteinspaltungen eine Identifizierung mittels Datenbanksuche nach einer der oben vorgestellten Varianten möglich, da für jedes Protein mindesten fünf Teilsequenzen mit 3 Aminosäuren erhalten wurden. Da zusätzlich für jedes Protein mindestens eine Teilsequenz mit mehr als 7 Aminosäuren erhalten wurde, konnte für jedes untersuchte Protein auch eine eindeutige Identifizierung über eine abschließliche Sequenzsuche erfolgen.

Ist das gesuchte Protein nicht selbst in einer der Protein- oder Nucleotiddatenbank enthalten, so kann beispielsweise eine Homologiesuche über BLAST erfolgen. Hierzu sollte die erhaltene Teilsequenz mindestens 7-8, besser jedoch 10 oder mehr Aminosäuren besitzen. Während mindestens eine **durchgehende** Sequenz mit mehr als 7 Aminosäuren für alle Proteine erhalten wurde, ergab sich für 10 der 16 untersuchten Proteinspaltungen auch mindestens eine Sequenz mit mehr als 10 Aminosäuren.

Tabelle 56: Resultate der Homologiesuche mit Teilsequenzen aus den LysC-Fragmenten

gesuchtes Protein	gesuchte Teilsequenz	AS	<i>E</i> (Expected ¹)	Trefferzahl ²	positive Homologien zum gesuchten Protein ³
Cytochrom C P00004	EYLENPK	7	10000	62	60
	TYTDANK	7	10000	5	4
Myoglobin P02188	YLDFISD	7	100000	100	94
	VLNVWGK	7	10000	65	65
β-Lactoglobulin P02754	VDDEALEK	8	10000	11	11
	YLLFCME	7	10000	16	16
β-Casein P02666	DMPIQAFLLYQEPVL	15	1000	19	19
	FQSEEQQTEDELQDK	15	10	12	12
Aldolase P00833	RALQASALK	9	10000	17	17
	AAQEYVK	8	10000	17	17
Rinderserum	YLYLIARR	8	10000	25	22
Albumin P02769	TVMENFVAFVDK	12	10000	31	19
	TCAVDESHAGCEK	13	1000	20	20
	VASLRETYGDMADCCEK	17	10	25	23

Als Suchmatrix wurde BLOSUM62 verwendet. Aufgrund der für die Homologiesuche recht kurzen Sequenzen aus der Leitersequenzierung muss der „Low Complexity“ Filter bei der Suche ausgeschaltet werden.

¹ Je kürzer die gesuchte Teilsequenz ist, desto höher muss der Wert für „Expected“ angegeben werden (Erklärung siehe nachfolgenden Text).

² Bezeichnet die Zahl aller vom Programm zurückgegebenen, homologen Sequenzen

³ Bezeichnet die Zahl der vom Programm zurückgegebenen Sequenzen, die wirklich homolog zum gesuchten Protein sind, also z.B. die Zahl aller Cytochrom Sequenzen bei der Suche mit den Cytochrom C Teilsequenzen.

Tabelle 56 zeigt, dass die Homologiesuche mit mindestens einer der erhaltenen Teilsequenzen für ein Protein in allen Fällen erfolgreich war. Die Datenbanksuche wurde über den BLAST Algorithmus durchgeführt (<http://www.ncbi.nlm.gov/blast>). Gesucht wurde in allen nicht-redundanten Datenbanken (595510 Sequenzen). Die mittels enzymatischer Leitersequenzierung generierten Teilsequenzen eignen sich in ihre Länge somit üblicherweise auch für Homologiesuchen.

Typischerweise muss der Wert für den Parameter E („Expected“) mit den vorliegenden, relativ kurzen Teilsequenzen von 7-12 Aminosäuren, sehr hoch gesetzt werden, damit die Datenbanksuche erfolgreich ist. Der benutzerdefinierte Parameter E ist ein Maß für die statistische Signifikanz der gefundenen, übereinstimmenden Sequenzen aus der Datenbank. Der Wert von E kann als diejenige Anzahl von Treffern angesehen werden, von der angenommen wird, dass sie bei einer Datenbanksuche mit einer Teilsequenz rein zufällig auftreten. Die Berechnung von E beim BLAST Algorithmus basiert auf der Karlin-Altschul Statistik [Karlin 1990] und erfolgt über den *Score* der HSPs (High-scoring Segment Pairs). Die HSPs stellen den grundlegenden Rückgabewert des BLAST Algorithmus bei der Suche mit einer Teilsequenz dar. Ein HSP besteht in Prinzip aus zwei Fragmenten (z.B. der eingegebenen Suchsequenz und einer Sequenz in der Datenbank) gleicher Länge, deren lokale Übereinstimmung maximal ist. Für den Wert E eines HSP gilt:

$$E = K \cdot N \cdot \exp(-\lambda \cdot S) \quad (12)$$

S : *Score* eines HSP

N : Produkt der Längen (Aminosäuren) von eingegebener Suchsequenz und Sequenz in der Datenbank

K, λ : Karlin-Altschul Parameter (positiv)

\exp : Exponentialfunktion

Hat also der Wert für den *Score* S eines HSP nur einen kleinen positiven, oder gar negativen Wert, so ist der zugehörige Wert für E hoch.

Auch molekularbiologisch ergeben sich aus den Sequenzdaten der Leitersequenzierung Anwendungsmöglichkeiten, sofern die Sequenzlänge ≥ 7 Aminosäuren ist, was bei allen Proteinspaltungen für mindestens eine Sequenz zutrifft. Diese Sequenzinformation erlaubt hier unter anderem bereits die Herstellung von Oligonucleotidsonden, die sich für die Detektion von Punktmutationen einsetzen lassen [Kessler 1998]. Aminosäuresequenzen mit 6-13 Aminosäuren liefern dabei die Information, die für die Herstellung der benötigten Oligonucleotide (üblicherweise 17-40 bp) erforderlich ist. Da der genetische Code stark degeneriert ist, d.h. viele Aminosäuren über mehr als nur ein Oligonucleotid-Triplett codiert werden, sollte die verwendete Leitersequenz möglichst viele Aminosäuren enthalten, die nur über wenige Codons spezifiziert werden. Von Vorteil für das Sondendesign sind daher besonders methionin- oder tryptophanhaltige Sequenzen als Ausgangspunkt, da Methionin und Tryptophan jeweils nur über ein einziges Codon spezifiziert werden. Gut geeignet sind des weiteren Teilsequenzen mit den Aminosäuren Glu, Asp, His, Lys, Gln, Asn, Cys, Phe, Tyr und Ile, während Sequenzen mit den Aminosäuren Ala, Val, Pro, Gly und insbesondere Arg, Ser und Leu weniger geeignet sind.

Auch für die Herstellung von degenerierten Primern zur Amplifikation mittels PCR lassen sich die Aminosäuresequenzen aufgrund ihrer Länge nutzen [Haberhausen 1998]. Auch hier sind Nucleotidsequenzen, wie sie sich aus Sequenzen von mindestens 6 Aminosäuren ergeben ausreichend. Bezüglich der Degenerierung gelten für die enthaltenen Aminosäuren in den Teilsequenzen die gleichen Präferenzen wie für das Sondendesign.

Auch mit der Edman-Sequenzierung lassen sich natürlich problemlos kurze Hexa- und Heptapeptidabschnitte sequenzieren, die wie gesehen den meisten Identifizierungsmöglichkeiten und Anwendungen genügen. Die entscheidenden Vorteile, die eine Leitersequenzierung dabei gegenüber einer Edman-Sequenzierung besitzt sind jedoch hohe Sensitivität, Schnelligkeit und geringe Kosten bei hohen Probenzahlen, insbesondere bei entsprechender Automatisierung. Zusätzlich besteht wie gesehen die Möglichkeit post-translationale Modifikationen direkt zu detektieren. Damit stellt sich in diesem Kontext sicherlich auch die Frage ob die Leitersequenzierung bei einer Proteinsequenzierungen mittels überlappender Spaltfragmente eine Alternative zur Edman-Sequenzierung darstellen kann. Von vier Proteinen wurden durch Spaltung mit Endopeptidasen unterschiedlicher Spezifität überlappende Spaltfragmente erzeugt.

Sequenziertes Protein	Spaltung mit folgenden Proteasen
β -Lactoglobulin	<ul style="list-style-type: none"> • Endoproteinase LysC • Endoproteinase GluC (Phosphatpuffer)
<i>DrTI (Serin Proteinase Inhibitor)</i>	<ul style="list-style-type: none"> • Endoproteinase LysC • Endoproteinase AspN
Aldolase	<ul style="list-style-type: none"> • Endoproteinase LysC • Endoproteinase GluC (Phosphatpuffer) • Trypsin • Chymotrypsin
Myoglobin	<ul style="list-style-type: none"> • Endoproteinase LysC • Trypsin

Die Resultate für Myoglobin sind hier ebenfalls dargestellt. Allerdings ist zu beachten, dass für dieses Protein nur eine Spaltung mit Endoproteinase LysC und Trypsin durchgeführt wurde. Diese Spaltungen führen lediglich beim Arginin, nicht aber beim Lysin zu überlappenden Spaltfragmenten.

Für die genannten Proteine ergeben sich aus den Resultaten unter Punkt 3.2.2 (Grafische Darstellung der Sequenzierungsergebnisse – Sequenzalignments, Seite 127) folgende Sequenzabdeckungen:

Tabelle 57: Sequenzabdeckungen durch überlappende Spaltfragmente

Protein	Zahl der Endopeptidase-Spaltungen	Sequenzabdeckung bezogen auf die Gesamtsequenz (Vergleichswerte für nur eine Endopeptidase)
β -Lactoglobulin	2	66% (23% - 56%)
<i>DrTI</i>	2	52% (16% - 45%)
Myoglobin	2	44% (24% - 35%)
Aldolase	4	80% (21% - 50%)

Die Daten zeigen, dass die Sequenzabdeckung bezüglich der Gesamtsequenz durch die Summe der Resultate zweier unterschiedlicher Endopeptidasespaltungen im Durchschnitt um ca. 10% ansteigt. Betrachtet man die überlappenden Spaltfragmente eines Protein, so spiegelt dieser relativ geringe Anstieg bereits eine schwache Überlappung der erhaltenen Teilsequenzen wider. Eine genauere Auswertung der Sequenzalignments für β -Lactoglobulin, *DrTI* (*Serin Proteinase Inhibitor*) und Myoglobin zeigt, dass für alle drei Proteine kaum überlappende Teilsequenzen vorhanden sind. Damit ist es trotz der hohen Sequenzabdeckung, beim Vorliegen zweier komplementärer Endopeptidasespaltungen in diesem Fall nicht möglich über die Abfolge der Teilsequenzen die Gesamtsequenz zu rekonstruieren.

Bei Aldolase wurde die Summe aus vier verschiedenen Endopeptidasespaltungen betrachtet. Hier beträgt die Sequenzabdeckung bezogen auf die Proteingesamtsequenz 80% und ist damit als ausgezeichnet zu beurteilen. Dennoch zeigt eine Detailauswertung des Alignments auch hier, dass eine Rekonstruktion der Gesamtsequenz nicht möglich ist. Es existieren 12 kurze Teilsequenzen im Protein, die nach keiner der vier durchgeführten Endopeptidasespaltungen durch Teilsequenzen abgedeckt werden. Somit wird auch hier zu häufig eine fehlende Überlappung einzelner Teilsequenzen beobachtet.

Erst unter der Voraussetzung, dass zusätzlich benötigte Exopeptidasen, vor allem Prolin Aminopeptidase (EC 3.4.11.9) auf N-terminaler Seite stehen, kann die Leitersequenzierung in Zukunft eine echte Alternative zur Edman-Sequenzierung darstellen. Im Vergleich zur *de novo* Sequenzierung auf rein massenspektrometrischen Weg (MS/MS Fragmentierungen, MALDI-PSD) stellt die Leitersequenzierung allerdings bereits heute eine leistungsfähige Methode im Bezug auf Schnelligkeit und Empfindlichkeit dar. Zudem ist die Interpretation der Leiterspektren als Resultat einer Sequenzierung wesentlich einfacher und damit schneller, als die Interpretation eines MS/MS Fragmentspektrums oder eines MALDI-PSD Spektrums. Analog ist auch die Möglichkeit der Proteinidentifizierung über kurze Sequenztags, die mittels Leitersequenzierung generiert wurden positiv im Vergleich zu rein massenspektrometrisch erzeugten Tags zu beurteilen.

Kapitel III:

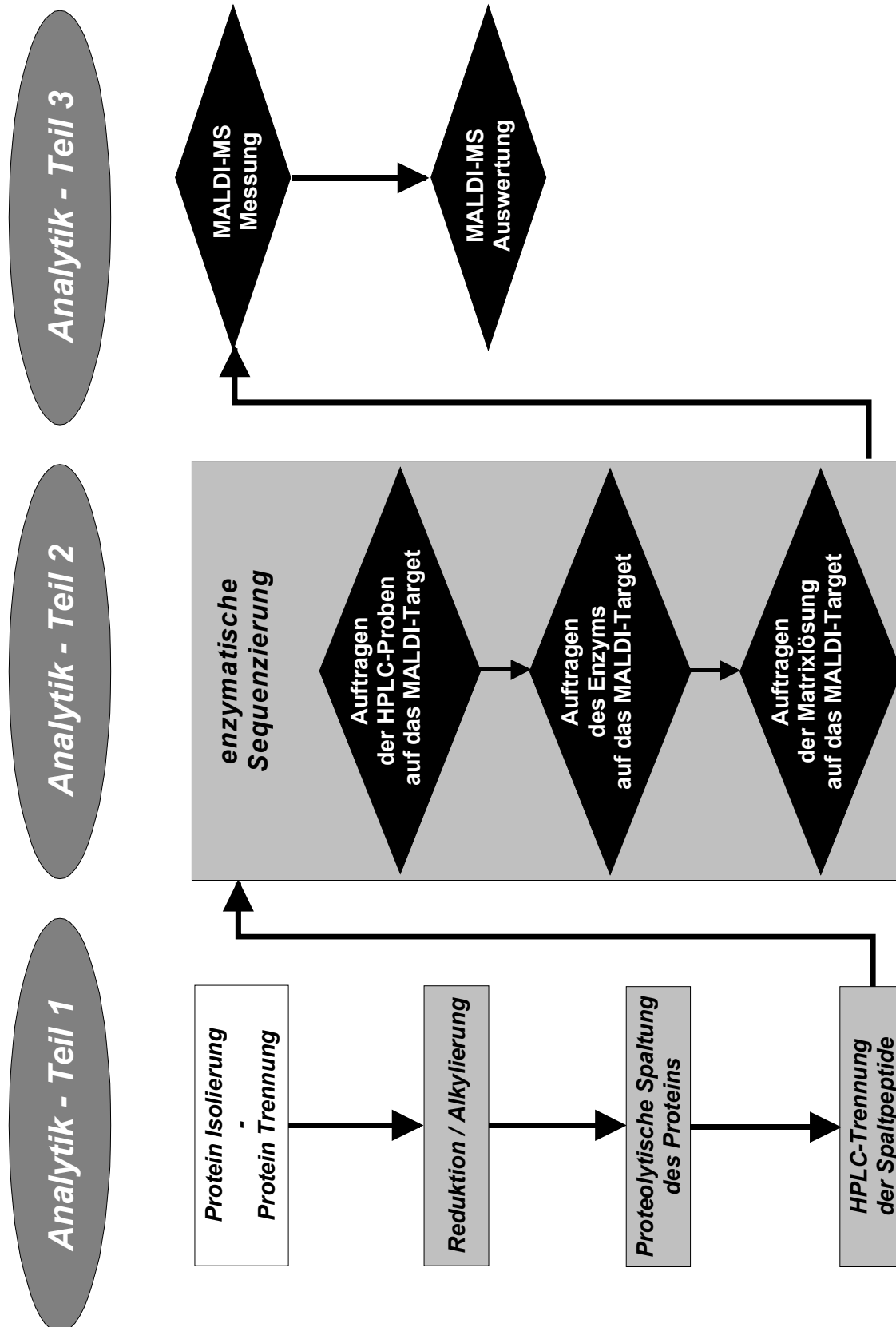
Automatisierung

1 Anforderungen an die Methode

Im vorangegangenen Kapitel wurden experimentelle Bedingungen ausgearbeitet, die eine enzymatische Leitersequenzierung und deren Anwendung auf proteolytische Spaltpeptide ermöglichen. Hierunter fielen unter anderem die Beurteilung verschiedener Exopeptidasen, Matrices oder Derivatisierungsreagenzien bezüglich ihrer Eignung für die Leitersequenzierung. Gerade für die Proteomanalytik besteht der Bedarf einer hochparallelen Sequenzierung zur Bearbeitung einer großen Anzahl von Proteinen, wie sie beispielsweise aus der 2-dimensionalen Gelelektrophorese erhalten werden. Im Hinblick auf eine Übertragung der erarbeiteten Methode(n) zur Leitersequenzierung auf einen solch parallelen Sequenzierungsansatz wurde daher in zweiten Teil dieser Arbeit eine mögliche Automatisierung einzelner Teilschritte der Methode(n) evaluiert.

Nach der gelelektrophoretischen Trennung von Proteinen und dem Ausschneiden der interessierenden Proteinspots erfolgt im Rahmen der Proteomanalyse die weitere Bearbeitung der einzelnen Proteinproben analog zu der im vorangegangenen Kapitel II eingehend diskutierten Sequenzierung über proteolytische Spaltfragmente. Ob dabei versucht werden soll, das Protein *de novo* zu sequenzieren oder nur möglichst viele *Sequenz-Tags* zu erhalten, sollte im Bezug auf die zu erarbeitende Automatisierung der Sequenzierung keine Rolle spielen. Für die geplante Umsetzung der Automatisierung wurden zunächst die einzelnen Schritte im Verlauf der Probenbearbeitung – vom intakten Protein bis hin zum Auslesen der Teilsequenzen aus dem Leiterspektrum – sinnvoll gegliedert. Eine solche Gliederung ist in Abbildung 72 dargestellt. Sie zeigt, dass sich nach der Isolierung der Proteine aus dem Gel eine Automatisierung der Analytik mehr oder weniger stark ausgeprägt für drei getrennte Teilbereiche ausarbeiten lässt. Im Rahmen dieser Arbeit liegt der Schwerpunkt für die Automatisierung auf dem Teilbereich 2. Für diesen Teilbereich der enzymatischen Sequenzierung sind die wesentlichen Einzelschritte, die eine Automatisierung im Ablauf erfordern, detaillierter aufgelistet. Automatisierte MALDI-MS Messungen sind mit den derzeit technisch zu Verfügung stehenden Mitteln ebenfalls bereits machbar. Der hohe Arbeits- und Zeitaufwand bei manueller Messung der in Teil 2 generierten Peptidleitern erfordert prinzipiell auch bei der MALDI-MS Messung dieser Proben zwingend die Nutzung einer solchen Automatisierung. Daher wurde auch untersucht, ob die aus dem analytischen Teilbereich 2 stammenden Leiterpeptidproben mit den zur Verfügung stehenden Mitteln auch eine automatische MALDI-MS Messung erlauben. Schwierigkeiten sind in diesem Fall bei der Umsetzung durch die Verwendung von DHB als Matrix zu erwarten.

Abbildung 72: Gliederung der Arbeitsschritte einer enzymatischen Leitersequenzierung ausgehend vom intakten Protein bis zur Leitersequenz



Eine automatische Abarbeitung der in Teil 1 der Analytik beinhalteten Schritte wurde in dieser Arbeit bewusst nicht untersucht, sollte jedoch für ein sinnvolles Gesamtkonzept der Sequenzierungsmethode in weiteren Arbeiten ebenfalls realisiert werden. Die Automatisierung bezüglich der Proteintrennung fällt im Rahmen von Proteomics unter den Teilaspekt der 2D-Gelelektrophorese. Anstrengungen diesen Bereich weitgehend zu rationalisieren und automatisieren sind bereits von mehreren Seiten im Gange. Die folgenden Schritte einer Reduktion und Alkylierung der Proteine nach deren Trennung, sowie die proteolytische Spaltung beinhalten für eine Automatisierung im wesentlichen reine Pipettierschritte. Eine Automatisierung erscheint hier generell unter den gleichen Voraussetzungen und Bedingungen möglich, wie sie im folgenden für den analytischen Teil 2 beschrieben werden. Auch die anschließende chromatographische Trennung der proteolytischen Spaltfragmente kann theoretisch bereits heute mit geeigneten Geräten zur automatischen Probeninjektion und intelligenter Software zur Peakfraktionierung in einen automatisierten Ablauf eingebunden werden. Auf Seiten der Peakerkennung innerhalb der verfügbaren Chromatographie-Software sind allerdings noch Verbesserungen wünschenswert und auch ein Interface zur Anbindung der Chromatographie an die Sequenzierung ist derzeit auf dem Markt kommerziell noch nicht verfügbar. Eine entscheidende Herausforderung besteht bei einer Automatisierung, die auch den Teilbereich 1 der Analytik nach Abbildung 72 umfassen soll, derzeit in der Gesamtintegration aller Schritte. Eine solche integrierende Lösung sollte auch die Datenerfassung (Probeneingabe), Datenweitergabe (Kommunikation zwischen den einzelnen Hardwarekomponenten) und die Datenauswertung umfassen. Die Automatisierung wurde im weiteren Verlauf zwar nur für die Teilbereiche 2 und 3 untersucht, jedoch wird eine effiziente Proteomanalytik in Zukunft nur dann möglich sein, wenn es gelingt, alle genannten Teilbereiche der Analytik mit einer Automatisierung zu erfassen. Jeder manuelle Schritt stellt einen „Flaschenhals“ im Gesamtsystem dar, der letztlich für den Probendurchsatz limitierend ist. Einige grundlegenden Anforderungen an die zu erarbeitende Automatisierung der Leitersequenzierung lassen sich wie folgt zusammenfassen:

- ❖ Einfache und reduzierte Dateneingabe (Bearbeitungsanweisungen)
- ❖ Flexibilität der Methode
- ❖ Schnelle, rationalisierte Probenabarbeitung
- ❖ Weitgehender Ausschluss manueller Eingriffe in den Ablauf

Ziel bei der Methodenerstellung sollte es sein, jede Probe auch individuell bearbeiten zu können. Das Abarbeitungsschema sollte es also beispielsweise erlauben, einen Teil der vorliegenden Proben C-terminal, den anderen Teil N-terminal zu sequenzieren oder einige spezielle Proben zu derivatisieren. Daher sollte es bei der Probenbearbeitung möglich sein, für jede Proben spezifische Bearbeitungsschritte zu definieren.

Der Punkt der Dateneingabe ist somit eng an die Flexibilität der erstellten Methode gekoppelt. Bei der Eingabe der Bearbeitungsinformationen für die Proben sollte ein hohes Maß an Flexibilität gegeben sein. Die Bearbeitung jeder Probe muss sich im Eingabedialog individuell definieren lassen. Dabei sollte jedoch auch die Möglichkeit einer vereinfachten und schnellen Dateneingabe für einzelne Arbeitsschritte bestehen, in denen jede Probe gleich behandelt wird (z.B. Matrixaufgabe auch alle Proben). Für die Eingabe der Bearbeitungsinformation der analysierenden Proben in das System sollte dem Benutzer ein möglichst einfaches Programminterface zur Verfügung gestellt werden, am besten in Form eines Formulars oder einer Eingabemaske. Idealerweise sollte die Eingabe der Probeninformationen nur einmal notwendig sein, wobei alle Proben mit den erforderlichen Informationen für Sequenzierung **und** MALDI-MS versehen werden. Beide Systeme (die Steuerung des Pipettiersystems und des MALDI-MS) sollten also das Datenformat der eingegebenen Bearbeitungsinformation einlesen können.

Auf der Seite der eigentlichen Sequenzierung kommt einer schnellen und rationalisierten Abarbeitung eine besondere Bedeutung zu. Die Pipettierschritte müssen im Programmablauf so angeordnet werden, dass Fahrwege oder Zwischenspülschritte weitgehend minimiert werden. Wichtig erscheint es im Hinblick auf den Zeitfaktor der Gesamtmethode (Pipettierschritte und MALDI-MS Messung) auch, dass manuelle Eingriffe im Ablauf der Probenabarbeitung stark minimiert werden. Der manuelle Eingriff in das Pipettiersystem sollte sich vor dem Programmstart der eigentlichen Sequenzierung auf das Aufstellen von Proben und Reagenzien auf der Oberfläche der Maschine beschränken. Erforderliche Verdünnungsschritte für Proben oder Reagenzien sollten bereits optionale Bestandteile des Programms sein. Nach Erledigung der Pipettieraufgaben für die Sequenzierung ist im optimalen Fall dann nur noch der (manuelle) Transfer der fertig präparierten Probenträger zum MALDI-MS notwendig.

2 Experimenteller Teil

2.1 Verwendete Proteine, Peptide, Enzyme und Chemikalien

Die im Rahmen der Versuche zur Automatisierung verwendeten Proteine, Peptide, Enzyme und Chemikalien entsprechen denen aus Punkt 2.1 in Kapitel II (Seite 66 ff.).

2.2 Verwendete Geräte

Zusätzlich zu den unter Punkt 2.2 (Kapitel II) aufgeführten Geräten wurden für die Automatisierung noch folgende Geräte verwendet:

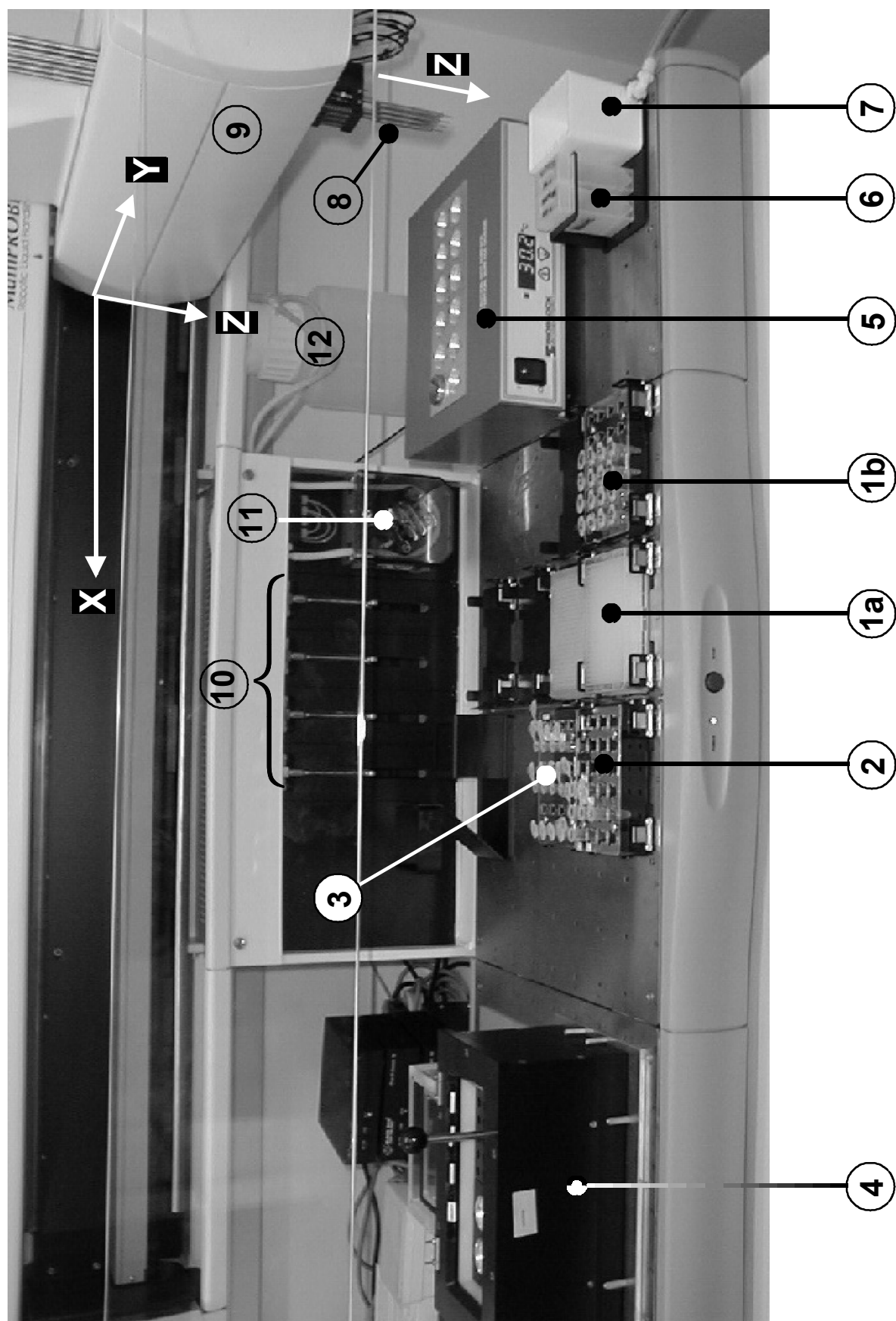
2.2.1 Automatisches Pipettiersystem

Als Pipettierrobotersystem wurde ein Multiprobe II der Firma Canberra Packard (Dreieich) verwendet. Für die enzymatische Sequenzierung und die Präparation der Proben wurde dabei mit folgender Gerätekonfiguration gearbeitet:

- Geräte-Layout: “*MultiPROBE II Extended System*” mit linkem und rechtem Expansionsmodul
- Pipettennadeln: Versa Tip mit nl-Option
- Spritzenassembly: 4x250µl Spritzen
- Probenhalterungen: - Microfuge Racks 0,5ml
- Microfuge Racks 1,5ml
- Heizblöcke: - Bioblock Scientific, BAIN A SEC 86207
Lab-Line Instruments, Welrose Park, USA
Einsatz für 12 MALDI-Targets mit SCOUT 26 Layout
- Heizblock aus den Werkstätten des
MPI für Biochemie, Martinsried
Einsatz für 24 Probengefäße 0,5ml (4x6 Layout)
Einsatz für 20 Probengefäße 1,5ml (4x6 Layout)Thermoblock

Ein Bild des Geräts mit dem Deck-Layout, wie es bei den hier besprochenen, automatischen Präparationen verwendet wurde, ist in Abbildung 73 auf der folgenden Seite dargestellt. Dabei sind auch die wichtigsten Merkmale des Geräts und Deck-Layouts hervorgehoben.

Abbildung 73: Verwendetes Pipettiersystem MultiPROBE II für die Automatisierung der Sequenzierung



Beschreibungen für Abbildung 73:

- 1a) 384er Mikrotiterplatte für Proben
- 1b) 0,5ml Polypropylengefäße für Proben
- 2) 1,5ml Polypropylengefäße für Stammlösungen von Matrices, Derivatisierungsreagenzien, Pufferlösungen, etc.
- 3) 0,5ml Polypropylengefäße für gebrauchsfertige Lösungen von Matrices, Derivatisierungsreagenzien, Puffersubstanzen, Enzymen und Kalibranden.
- 4) Thermoblock (Werkstätten MPI, siehe oben) zur Temperierung von Enzymen in 0,5ml Polypropylengefäßen und 4 MALDI-Targets (Typ: SCOUT 26)
- 5) Thermoblock (Bioblock Scientific, BAIN, siehe oben) zur Temperierung von bis zu 12 MALDI-Targets (Typ: SCOUT 26)
- 6) Waschflüssigkeit (Acetonitril/Wasser 50/50 (v/v) + 0,1% TFA) zur Reinigung der Pipettiernadeln
- 7) Waschgefäß zur Reinigung der Pipettiernadeln mit Systemflüssigkeit (deionisiertes, entgastes Wasser).
- 8) Pipettiernadeln (4 Stück); nl-Spitzen für Volumina bis minimal 0,1µl
- 9) Arm für die Bewegung der Pipettiernadeln in x-, y- und z-Richtung
- 10) Spritzenpumpen mit 250µl Volumen zum Pipettieren aller Lösungen
- 11) Peristaltikpumpe zur Reinigung des Systems mit Systemflüssigkeit
- 12) Systemflüssigkeit (Vorratsbehälter)

2.2.2 Massenspektrometer

Die Spezifikationen entsprechen denen des unter Kapitel II beschriebenen MALDI-Massenspektrometers „Bruker Reflex III“ (siehe S. 75) der Firma Bruker Daltonik (Bremen). Alle Untersuchungen zur Automatisierung beschränken sich in dieser Arbeit auf UV-MALDI-MS

2.3 Verwendete Softwarekomponenten

2.3.1 Automatisches Pipettiersystem

- WinPREP for Multisource II Canberra Packard, Dreieich

Die Eingabe des Programms kann wahlweise über die grafische Oberfläche der WinPREP Software oder durch direkte Eingabe des Quellcodes erfolgen. Die dem Programm WinPREP zugrunde Programmiersprache ist „*Multiprobe Script Language*“ (MSL). Diese Programmiersprache ist sehr eng an C / C+ angelehnt und unterstützt den Großteil des C-Befehlssatzes. MSL enthält jedoch als spezielle Programmiersprache für Pipettieraufgaben einige zusätzlich Befehle für das *Liquid Handling*. Alle Schritte eines Programms, das über die grafische Oberfläche der WinPREP Software eingegeben wurde, werden vor dessen Ausführung in MSL übersetzt. Die Ausarbeitung eines Programms kann sehr flexibel gestaltet werden, denn auch innerhalb der grafischen Oberfläche besteht die Möglichkeit fehlende, aber für den Programmablauf notwendige Anweisungen durch die direkte Eingabe von MSL-Quellcode zu ergänzen. Von dieser Möglichkeit wurde an einigen Stellen des ausgearbeiteten Programmablaufs Gebrauch gemacht.

2.3.2 Automatische MALDI-MS Messungen

- AutoXecute™ 4.2 Bruker Daltonik, Bremen

Die Programme für Primärakquisition auf OS-9, Akquisition und Auswertung auf UNIX entsprechen den in Kapitel II aufgelisteten Komponenten unter 2.2.2 (Seite 75).

2.3.3 Zusätzliche Softwarekomponenten

- Microsoft Excel 97 Microsoft Corporation, USA
- FTP Control Version 3.81 TransSoft Ltd., Niederlande
- Microsoft Access 97 Microsoft Corporation, USA

3 Ergebnisse

Die selbst erarbeiteten und im Text namentlich erwähnten Programmdateien für die Automatisierung der Sequenzierung und der MALDI-MS Messungen befinden sich zum größten Teil aus Platzgründen auf der im Anhang E befindlichen CD-ROM. Alle erforderlichen Programmdateien liegen hier im Originalformat vor und sind so auch mit den eingesetzten Hard – und Softwarekomponenten voll funktionsfähig. Sofern die Dateien nicht binär vorliegen, sondern sich in einem Textformat (ASCII) darstellen lassen, sind diese in gedruckter Form gemäß §12 Abs. 6 der Promotionsordnung vom 29. Januar 1998 als getrennter Anhang beigelegt. Auf der beiliegenden CD können diese Dateien mit jedem zur Verfügung stehenden Texteditor eingesehen werden. Eine detaillierte Beschreibung und Benutzungsanleitung der beiliegenden CD findet sich ebenfalls in Anhang E.

3.1 Die Sequenziermethode „*MALDI-LS 1.42s*“ für den MultiPROBE II

3.1.1 Allgemeines zur Sequenziermethode „*MALDI-LS 1.42s*“

Die komplette Methode, wie sie mit der Software WinPREP und zusätzlichen MSL-Funktionen unter Windows NT 4.0 erstellt wurde, einschließlich kompletter Testinformationen („*Deck View*“, „*Test Outline*“, „*Parameters*“, „*Well Map*“) und Systeminformationen („*Labware Definitions*“, „*Performance Tables*“, „*Instrument Settings*“) befindet sich in ihrer in seiner aktuell ausgearbeiteten Version 1.42s als Datei(en) auf der CD, die dieser Arbeit beiliegt. Die mit der Software WinPREP erstellte Methode ist auf der beiliegenden CD unter dem Dateinamen „*MALDI-LS 1.42s.MPT*“ zu finden. Auf einige Punkte der Methode wird in den folgenden Abschnitten noch genauer eingegangen, insbesondere auf den „*Performance File*“ des Programms für die Flüssigkeitsaufnahme und –abgabe beim Pipettierprozess und ebenso auf einige Funktionsaufrufe innerhalb des Programms. Zur Erstellung und Ausführung des Programms wurde zum Großteil die grafische Oberfläche des WinPREP verwendet. Die tatsächliche Ausführung des Programms erfolgt durch den MSL-Script Prozessor. Dieser gibt das über den grafischen Umweg generierte Skript an den MultiPROBE II weiter. Das eigentliche Programm liegt in der „MultiPROBE Script Language“ (MSL) vor (siehe oben). Die notwendigen MSL Dateien der Methode *MALDI-LS 1.42s.MPT* für den MSL-Script Prozessor sind *MALDI-LS 1.42s.s* , *WinPREP.s* und *MALDI-LS 1.42s.pro*. Das primär aus *MALDI-LS*

1.42s.MPT generierte Programm ist dabei *MALDI-LS 1.42s.s*. *WinPREP.s* ist ein für jede Methode benötigtes Standardskript. Zusammengefasst sind die benötigten Skripte in der Projektdatei *MALDI-LS 1.42s.pro*. Diese Projektdatei wird zur Ausführung aufgerufen und bringt dann letztlich die einzelnen Skripte im Projekt zur Ausführung. Die für die Methode *MALDI-LS 1.42s* spezifischen und entscheidenden Dateien *MALDI-LS 1.42s.s* und *MALDI-LS 1.42s.pro* befinden sich auch in gedruckter Form im getrennten Anhang.

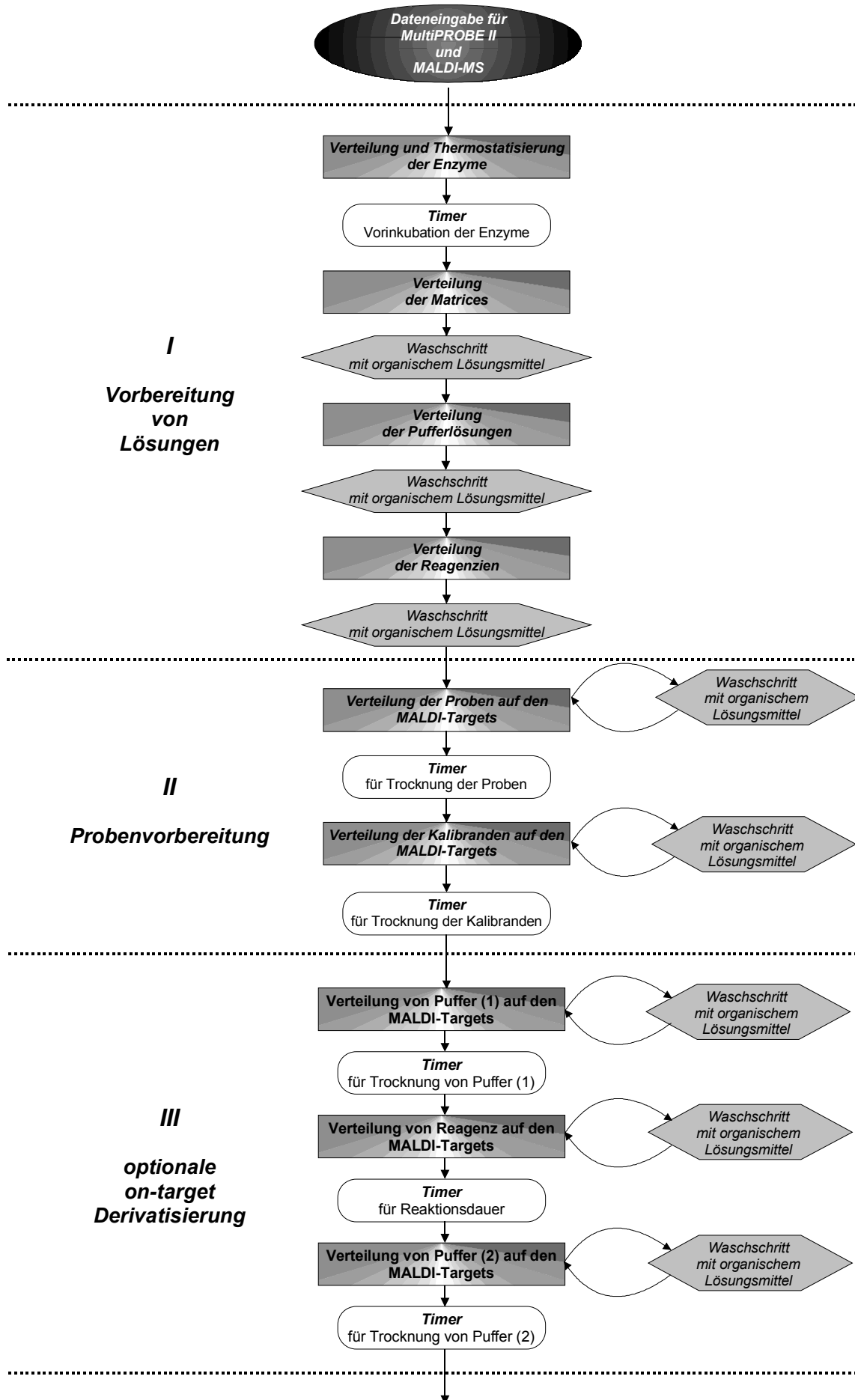
3.1.2 Arbeitsschritte in der Sequenziermethode „*MALDI-LS 1.42s*“

Es ist aus Übersichtsgründen nicht möglich und sinnvoll, alle einzelnen Arbeitsschritte der Methode *MALDI-LS 1.42s* aufzuführen. Das Schema in Abbildung 74 soll einen Überblick geben, welche Möglichkeiten die vorliegende Version der Methode für die Sequenzierung überhaupt bietet. Die gezeigte Gliederung in Abbildung 74 bezieht sich nur auf den Sequenzierungsteil und stellt damit einen detaillierteren Einblick in Teil 2 der Gesamtanalytik (Abbildung 72) dar. Es handelt sich um eine stark vereinfachte Darstellung des tatsächlichen „*Test Outline*“ des WinPREP-Programms. Man kann eine Gliederung der vorgestellten Methode formal in 5 Abschnitte vollziehen:

- I) Vorbereitung der Lösungen (Enzyme, Matrices, Reagenzien)
- II) Probenvorbereitung (Probenauftrag auf MALDI-Targets)
- III) optionale Möglichkeit zur Derivatisierung (*on-target*)
- IV) Enzymatische Sequenzierung (*on-target*)
- V) Auftragung der Matrix / Matrices

Abbildung 75 zeigt das für die Methode *MALDI-LS 1.42s* gewählte Basis-Decklayout auf dem MultiPROBE II. Die einzelnen Elemente, wie z.B. Heizblöcke, Reagenzgefäßhalter oder Tröge für Waschflüssigkeit sind dabei jedoch weder auf eine bestimmte Position fixiert, noch in ihrer Anzahl limitiert. Die Definition der Positionen erfolgt allein über die Bezeichnung der Elemente in der Methode. Die zu bearbeitende Probenanzahl ist lediglich über das Aufnahmevermögen des Systems limitiert, nicht über die erstellte Methode !

Abbildung 74 (folgende zwei Seiten): Schematische Darstellung der Abläufe bei automatischer Sequenzierung mit dem Programm *MALDI-LS 1.42s* auf dem MultiPROBE II



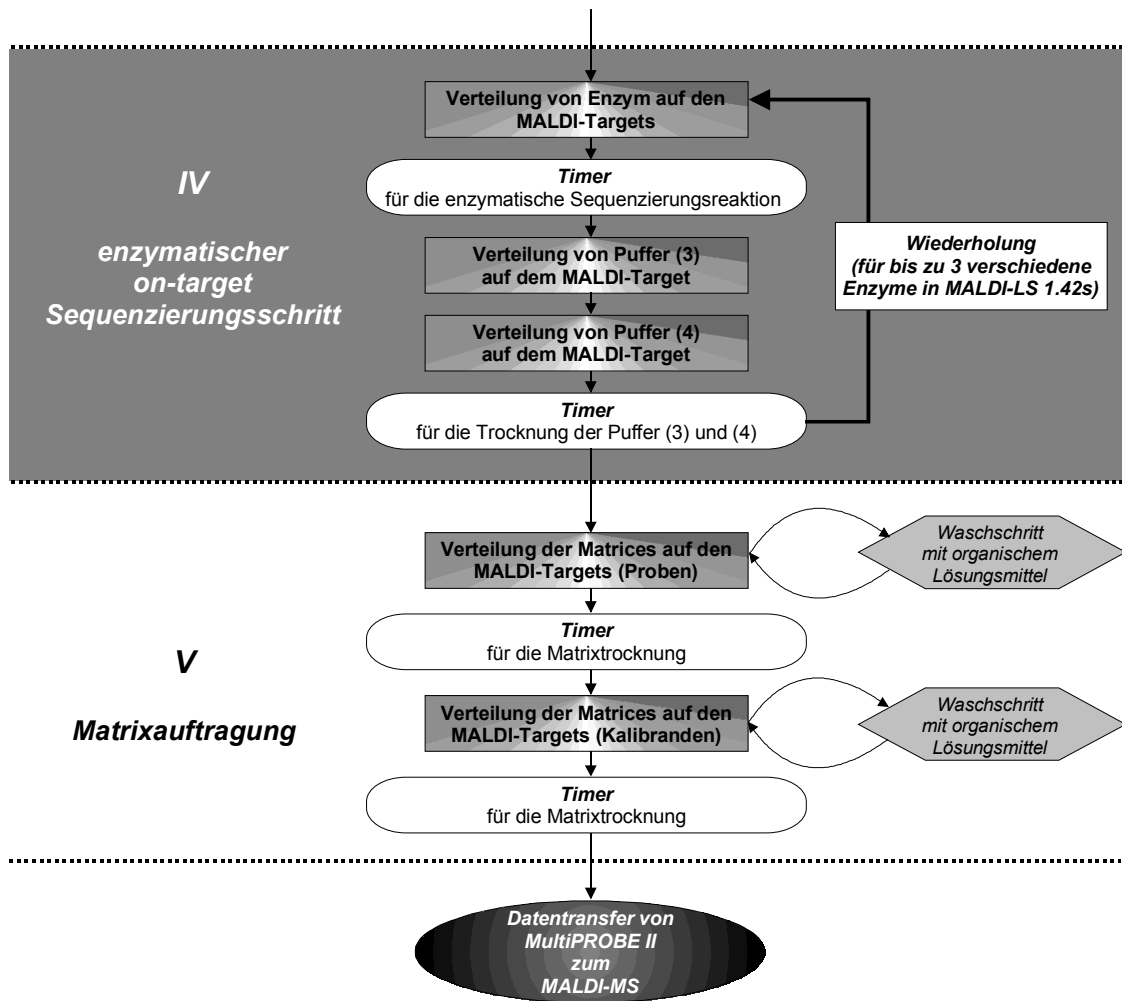
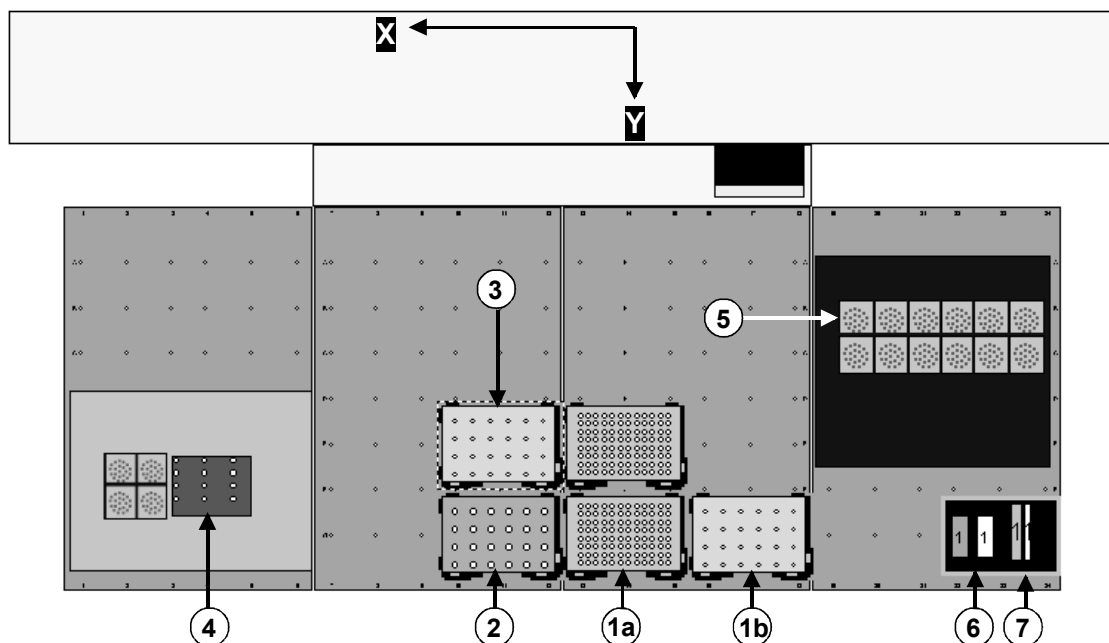


Abbildung 75: Schematische Darstellung des Basis-Decklayouts auf dem MultiPROBE II für die automatische Sequenzierungsmethode (Blick von oben in z-Richtung)



- 1a) 384er (oder 96er) Mikrotiterplatte für Proben
Bezeichnung in der Methode: „384 Samples #“ oder „96 Samples #“
 („#“: fortlaufende Nummer der Halterungen für die Probenvials)
- 1b) 0,5ml Polypropylengefäße für Proben
Bezeichnung in der Methode: „Eppendorf 0.5 Samples #“
 („#“: fortlaufende Nummer der Halterungen für die Probenvials)
- 13) 1,5ml Polypropylengefäße für Stammlösungen von Matrices, Derivatisierungsreagenzien, Pufferlösungen, etc.
Bezeichnung in der Methode: „Eppendorf 1.5 SSMBR“
 („SSMBR“: Stock Solutions of Matrices, Buffers and Reagents)
- 14) 0,5ml Polypropylengefäße für gebrauchsfertige Lösungen von Matrices, Derivatisierungsreagenzien, Puffersubstanzen, Enzymen und Kalibranden.
Bezeichnung in der Methode: „Eppendorf 0.5 EMC“
 („EMC“: Enzymes, Matrices and Calibrands)
- 15) 0,5ml Polypropylengefäße für die Temperierung der Enzyme
Bezeichnung in der Methode: „Eppendorf 0.5 HUE“
 („HUE“: Heating Unit for Enzymes)
- 16) MALDI-Target (temperiert)
Bezeichnung in der Methode: „MALDI #“
 („#“: fortlaufende Nummer des MALDI-Targets im Thermoblock; Nummerierung in Pfeilrichtung von links oben nach rechts unten)
- 17) Waschflüssigkeits- und Abfallbehälter (Reinigung der Pipettiernadeln mit ACN/Wasser-Gemisch)
Bezeichnung in der Methode: „Flush/Wash2“
- 18) Waschflüssigkeits- und Abfallbehälter (Reinigung der Pipettiernadeln mit Systemflüssigkeit)
Bezeichnung in der Methode: „Flush/Wash1“

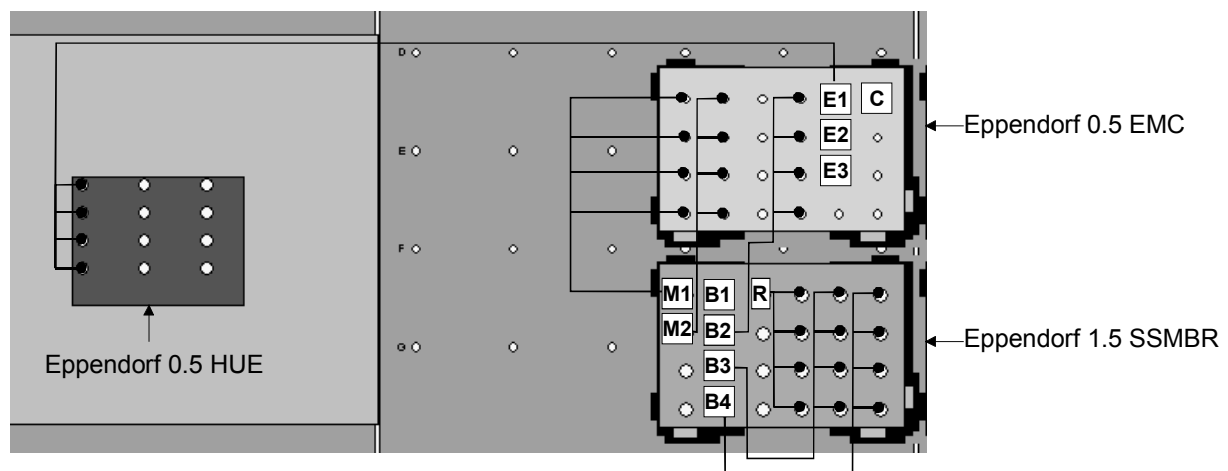
3.1.3 Einzelaspekte der Methode „MALDI-LS 1.42s“

3.1.3.1 Belegung des MultiPROBE II mit unterschiedlichen Racks

Im Rahmen eines Tests mit der Methode MALDI-LS 1.42s für Proben sind die Elemente mit den Bezeichnungen „384 Samples #“, „96 Samples #“ oder „Eppendorf 0.5 Samples #“ entscheidend (Positionen 1a und 1b in Abbildung 75). Bei ersteren beiden handelt es sich um Mikrotiterplatten im 384er oder 96 Format, während „Eppendorf 0.5 Samples #“ einen Ständer für 0,5ml Reaktionsgefäße darstellt. Stammlösungen von Matrices, Puffersubstanzen oder Derivatisierungsreagenzien finden auf Positionen des Elements „Eppendorf 1.5 SSMBR“ (Abbildung 75, ②) Platz. Es handelt sich um einen Ständer für 1.5ml Reaktionsgefäße. Hier wird von jeder benötigten Lösung ein größerer Vorrat bereitgestellt. Die zur Verfügung stehenden Volumina sind theoretisch für MALDI-Präparationen von über 1000 Proben aus-

reichend, und genügen somit auch späteren Anforderungen zur Bearbeitung größerer Probenzahlen. Stammlösungen der Enzyme, sowie eine Lösungen mit Kalibranden finden auf Positionen von „Eppendorf 0.5 EMC“ Platz (Abbildung 75, ©). Für Pilotstudien zur Automatisierung im Rahmen dieser Arbeit wurden im Einzelfall maximal 50 Proben sequenziert. Hierfür sind Enzymvolumina bis 500µl vollkommen ausreichend. Im Hinblick auf ein höheres Probenaufkommen ist eine einfache Adaption der Volumina jedoch möglich. Auch für die Kalibrandenlösung wurde nur ein Volumen von 500µl vorgelegt. Da die Kalibranden ohnehin für jedes MALDI-Target nur einmal aufgetragen werden (1µl pro Target), genügt ein maximales Volumen von 500µl auch den Anforderungen bei größeren Probenzahlen. Abbildung 76 zeigt einen Detailausschnitt von Abbildung 75 mit den Positionen der Stammlösungen auf den Elementen „Eppendorf 1.5 SSMBR“ und „Eppendorf 0.5 EMC“.

Abbildung 76: Detailausschnitt des Basis-Decklayouts aus Abbildung 75: Aufstellung und Verteilung von Stammlösungen in „MALDI-LS 1.42s“



- M1, M2:** Stammlösungen der Matrices (üblicherweise: M1 = DHB, M2 = CHCA)
B1, B2, B3, B4: Stammlösungen der Puffersubstanzen
R: Stammlösung des Derivatisierungsreagenz
E1, E2, E3: Stammlösungen der verschiedenen Enzyme
C: Lösung mit Kalibranden

3.1.3.2 Vorbereitung der Lösungen – Verteilungsschritte (I)

Viele Proben werden mit der gleichen Lösung einer Matrix, eines Puffers, eines Reagenz oder auch eines Enzyms behandelt. Die technische Ausführung des Pipettiersystems mit 4 parallel in y-Richtung angeordneten Nadeln (siehe Abbildung 73, ⑧) lässt einen Pipettiervorgang aus nur einem Gefäß aus Sicht des Zeitfaktors uneffizient erscheinen. Um dieses Problem zu umgehen, erfolgen zunächst für alle Stammlösungen Verteilungsschritte (Teilbereich I in Abbildung 74 „Vorbereitung der Lösungen“). Dabei wird aus der jeweiligen Stammlösung heraus das für die kommenden Arbeitsschritte benötigte Gesamtvolumen auf 4, in y-Richtung angeordnete Reaktionsgefäße verteilt. Dadurch kann im weiteren Arbeitsablauf mit allen 4 Nadeln gleichzeitig die benötigte Lösung aufgenommen und weiterverteilt werden. Abbildung 76 zeigt als Zusatzinformation, wie diese Verteilungsschritte Stammlösungen auf andere Reaktionsgefäße von „Eppendorf 1.5 SSMBR“, „Eppendorf 0.5 EMC“ und „Eppendorf 0.5 HUE“ ablaufen (Verteilungsweg symbolisiert durch —●). Für die Verteilung der Enzyme ist nur der Weg von E1 dargestellt – die Verteilung von E2 und E3 erfolgt analog auf „Eppendorf 0.5 HUE“. Die Enzyme werden gleichzeitig mit dem Verteilungsschritt in einen Thermoblock pipettiert und können so auch vortemperiert werden.

3.1.3.3 Probenverschleppung und sonstige Kontaminationen

Abbildung 74 zeigt, dass an sehr vielen Stellen der Methode Waschschrte mit organischem Lösungsmittel eingebaut sind. Diese Waschschrte sind notwendig, da bei allen Präparationen mit Festspitzen pipettiert werden muss. Aus Sicht einer kontaminationsfreien Präparation wäre sicherlich die Verwendung abwerfbarer Spitzen vorzuziehen. Allerdings bieten nur die verwendeten Festspitzen eine extrem hohe Genauigkeit beim Pipettieren geringer Volumina bis herunter zu 0,5µl. (Der Hersteller gibt für 0,5µl in der Spezifikation einen Fehler von 10% an.) Als geeignete Maßnahme zur Vermeidung von Probenverschleppung, Kontamination oder der Deaktivierung der Enzyme, sind bei der Verwendung von Festspitzen zum Flüssigkeitstransport zwischengeschaltete Spülschritte mit einem Lösungsmittel mit organischem Anteil unumgänglich, um Probleme insbesondere mit hydrophoben Peptiden, Reagenzien oder Matrices zu vermeiden. Rekursive Spülschritte mit organischem Lösungsmittel sind daher vor allem nach jedem Transportschritt für Proben und Matrices notwendig. Für die Präparationen in dieser Arbeit wurde ein Gemisch aus Acetonitril/Wasser 50/50 (v/v) zum Zwischenspülen verwendet (Position ⑥, Abbildung 75). Der Spülschritt mit organischem

Lösungsmittel erfolgt immer zwischen zwei weiteren Spülschritten mit Systemflüssigkeit. Bei der Systemflüssigkeit handelt es sich um Helium-entgastes, deionisiertes Wasser. Jeder Spülschritt wäscht die Nadel zunächst innen, dann innen und außen mit jeweils 2ml Systemflüssigkeit. Vor dem organischen Spülschritt wird somit weitgehend vermieden, dass die organische Waschflüssigkeit zu stark kontaminiert wird. Im Spülschritt mit dem organischen Lösungsmittel werden anschließend 10µl Flüssigkeit aufgenommen. Die Waschflüssigkeitsmenge entspricht ca. dem 7 bis 20-fachen der üblicherweise auf das Target pipettierten Flüssigkeitsmengen (1,5-0,5µl) und ist damit zum Reinigen des Nadelinneren ausreichend. Um die Nadel auch auf der Außenseite zu reinigen, wird vor der Aufnahme des organischen Lösungsmittels 5mm unter die Flüssigkeitsoberfläche eingetaucht. Diese Eintauchtiefe genügt, um die Nadel außen zu reinigen, da bei allen anderen Pipettierschritten nur der vordere Millimeter der Nadel kontaminiert wird. Die Flüssigkeitsaufnahme erfolgt in diesen Fällen nach einer Detektion der Flüssigkeitsoberfläche durch Leitfähigkeitsmessung („*liquid level sensing*“). Nach der Detektion der Flüssigkeitsoberfläche taucht die Nadel nur 1mm unter die Oberfläche und nimmt dort die Flüssigkeit auf.

In der Praxis hat sich gezeigt, dass durch die geschilderte Abfolge beim Spülen Probenverschleppungen nicht beobachtet werden konnten. Sowohl in Teil II „*Probenvorbereitung*“ des Programmablaufs von *MALDI-LS 1.42s*, als auch in Teil V „*Matrixauftragung*“ sollten Arbeitsschritte, welche die Kalibranden betreffen erst nach Bearbeitung der Proben erfolgen. Da die Kalibriersubstanzen in der Regel eine sehr hohen MS-Response besitzen, ist die Gefahr einer später sichtbaren und somit störenden Probenkontamination andernfalls zusätzlich erhöht. Gerade wenn in den zu untersuchenden Proben nur sehr geringe Konzentrationen vorliegen, muss selbst bei minimaler Verschleppung der Kalibranden mit starken Suppressionseffekten gerechnet werden.

3.1.3.4 Verwendung der Pufferschritte 1 bis 4

Die Methode verwendet bis zu vier verschiedene „Pufferlösungen“. Die Anwendung dieser Lösungen im Rahmen der Methode *MALDI-LS 1.42s* sind wie folgt vorgesehen:

- ❖ Puffer (1): Pufferungsschritt vor einer eventuellen Peptidderivatisierung zur Einstellung der notwendigen Reaktionsbedingungen;
- ❖ Puffer (2): Pufferungsschritt nach der optionalen Peptidderivatisierung zur Umstellung der Reaktionsbedingungen (z.B. pH-Wert) für den ersten enzymatischen Sequenzierungsschritt;

- ❖ Puffer (3): Schritte mit Puffer (3) sind für eine eventuelle Verlängerung der Spaltungsreaktion mit den zuvor aufgetragenen Enzym vorgesehen. Dazu sollte für Puffer (3) in der Regel Wasser verwendet werden oder eine Pufferlösung, die mit allen verwendeten Enzymen kompatibel ist.
- ❖ Puffer (4): Schritte mit Puffer (4) sind ebenfalls für eine eventuelle Verlängerung der Spaltungsreaktion vorgesehen. Hier kann ein anderes Puffersystem verwendet werden, sofern bei Verwendung von Enzymmischungen nicht für alle aufgetragenen Enzyme mit Puffer (3) eine optimale Abspaltungscharakteristik möglich ist. Zusätzlich erlaubt dieser Schritt auch eine Umpufferung für die Auftragung eines weiteren Enzyms bei sequenzieller Auftragung. Beispielsweise kann für Puffer (3) ein basisches und Puffer (4) ein saures System gewählt werden.

3.1.3.5 Kalibrierung

Als Standard zur Kalibrierung wurden 0,5µl einer äquimolaren Mischung der Peptide Angiotensin II (1046,5423 Da), Bradykinin (1060,5690 Da), Angiotensin I (1296,6853 Da), Neurotensin (1672,9176 Da) und ACHT 18-39 (2466,2022 Da) aufgetragen (alle Massenangaben $[M+H]^+$ monoisotopisch). Die Konzentrationen der einzelnen Peptide in der Mischung lagen im Bereich von 100-500fmol/µl je Peptid.

3.1.3.6 Parameter der Flüssigkeitsaufnahme und -abgabe

Für die Aufnahme und Abgabe der Flüssigkeiten beim Pipettierprozess müssen bestimmte Parameter festgelegt werden. Diese Parameter richten sich nach der zu pipettierenden Flüssigkeit und ihren speziellen Eigenschaften, wie z.B. Viskosität, aber auch nach dem zu pipettierenden Volumen. Wichtige Einflussgrößen, die dabei zu definieren sind, sind Aufnahme und Abgabegeschwindigkeit der Flüssigkeit oder die Volumenkompensation beim Flüssigkeitsausstoß. Unterschiedliche Flüssigkeiten, wie z.B. wässrige Lösungen, Serum oder DMSO benötigen hier zum Teil sehr spezifische Parameter. Diese Parametersätze sind für den Pipettierroboter in sogenannten „Performance Files“ zusammengefasst. Im Zusammenhang mit der enzymatischen Sequenzierung – ohne Berücksichtigung potentieller Derivatisierungsreaktionen – konnten die Optimierungen der Parameter auf wässrige Lösungen beschränkt werden. Diese Parameter ließen sich auch für Lösungen mit einem Acetonitrilgehalt bis maximal 50% anwenden. Neben der Verwendung fester Pipettierspitzen stellt auch der sogenannte „Blowout Modus“ eine bei der Flüssigkeitsabgabe eine notwendige Voraussetzung dar,

um Volumina im Sub- μ l-Bereich so genau wie möglich handhaben zu können. In diesem Modus wird vor der Aufnahme der Flüssigkeit ein definiertes Luftvolumen angesaugt. Bei der anschließenden Flüssigkeitsabgabe sorgt der Ausstoß dieses zusätzlichen Luftvolumens für eine exakte, vollständige Flüssigkeitsabgabe. Die Größe des *Blowout-Volumens* hat im Zusammenhang mit der Abgabegeschwindigkeit einen starken Einfluss auf die Güte der Flüssigkeitsabgabe auf dem MALDI-Target. Im Zusammenhang mit der Flüssigkeitsabgabe auf das MALDI-Target kommt erschwerend hinzu, dass der Ort der Abgabe planar ist und nur einen Durchmesser von 2,0mm besitzt. Zudem besteht keine direkte Abgrenzung der einzelnen Abgabeorte verschiedener Proben auf dem MALDI-Target. Der mittlere Abstand zweier Spots voneinander beträgt im Mittel gerade einmal 5mm. Bei der Flüssigkeitsabgabe ist also höchste Präzision gefordert um eine Kontamination, z.B. durch Verspritzen der Flüssigkeit bei der Abgabe, auf jeden Fall zu vermeiden.

Für alle Flüssigkeitsabgaben, die nicht das MALDI-Target betreffen, konnten ohne Probleme die vom Hersteller mitgelieferten *Performance Files* für wässrige Lösungen im *Blowout Modus* verwendet werden. Für die Flüssigkeitsabgaben auf dem MALDI-Target erwiesen sich diese *Performance Files* jedoch als nicht brauchbar. Da sich die Flüssigkeitsabgaben auf dem MALDI-Target auf Volumina von 0,5 μ l und 1,0 μ l beschränkten, wurden im *Performance File* die Parameter für diese beiden Volumina optimiert. Die folgende Tabelle zeigt die Resultate:

Tabelle 58: Entscheidende Einträge in den *Performance File* im Blowout-Modus bei 0,5 μ l und 1,0 μ l – gegenübergestellt für die Standardprobenabgabe und die Probenabgabe auf dem MALDI-Target

Eintrag im <i>Performance File</i>	0,5 μ l MALDI	0,5 μ l Standard	1,0 μ l MALDI	1,0 μ l Standard
Abgabegeschwindigkeit [μ l/s]	50	250	50	200
Abgabe Verzögerung [ms]	800	800	500	250
Blowout Volumen [μ l]	3,0	4,0	3,0	8,0
Blowout Verzögerung [ms]	0	0	0	0

Die Versuche ergaben, dass unter allen Parametern in den *Performance Files* lediglich die Abgabegeschwindigkeit und das *Blowout*-Volumen besonders kritische Größen bei der Flüssigkeitsabgabe auf das MALDI-Target sind. Mit den vorgegebenen Standardparametern die in Tabelle 58 ebenfalls dargestellt sind, erfolgt auf dem MALDI-Target eine zu unregelmäßige Verteilung des abgegebenen Volumens. Insbesondere die hohe Abgabegeschwindigkeit bei den Standardparametern sorgt dafür, dass die Flüssigkeit förmlich auf den Spot

„geblasen“ wird und dabei verspritzt. Für eine „saubere“ Flüssigkeitsabgabe auf dem MALDI-Spot ist mit 50 $\mu\text{l/s}$ eine gegenüber den Standardwerten deutlich niedrigere Abgabegeschwindigkeit zu wählen. Auch das *Blowout*-Volumen muss kleiner gehalten werden. Bei einem *Blowout*-Volumen über 3,0 μl zeigte sich eine zunehmend schlechtere Ortsschärfe des abgegebenen Flüssigkeitstropfens auf dem MALDI-Target.

Bei der Definition der MALDI-Target Geometrie für den Pipettierroboter muss als wichtige Größe auch die Abgabehöhe angegeben werden. Diese Größe gibt an, wie viel Millimeter über der Oberfläche des MALDI-Targets die Abgabe der Flüssigkeit erfolgen soll. Im Rahmen der Untersuchungen zeigt sich, dass die Abgabehöhe nur minimal über der Oberfläche liegen darf. In der Regel wurde der geringste Abstand von 0,1mm (kleinste Motorschrittweite) über Target-Oberfläche als Abgabehöhe eingestellt.

Die hier vorliegende Problematik der Flüssigkeitsabgabe stellt keinen Spezialfall für die verwendete MALDI-Target Geometrie (SCOUT26) dar. Auch bei MALDI-Targets anderer Hersteller und vor allem bei den mittlerweile erhältlichen Targets im 384er Mikrotiterplattenformat, kommt der Optimierung einer exakten Flüssigkeitsabgabe eine wichtige Bedeutung zu.

3.1.3.7 Spezielle Laufzeitvariablen im Programmablauf

Unter den speziellen Laufzeitvariablen können zwei Klassen unterschieden werden:

- ❖ Variablen, die Volumina definieren
- ❖ Variablen, die Zeitintervalle definieren

Die Laufzeitvariablen erhalten ihre Werte aus speziell geschriebenen MSL Funktionen. Diese MSL Funktionen wiederum berechnen die Werte der Variablen zumeist aus der Anzahl der zu bearbeitenden Proben. Tabelle 59 und Tabelle 60 zeigen einen Überblick über die in der Methode *MALDI-LS 1.42s* enthaltenen Laufzeitvariablen, ihre Position in der Methode und ihre Bedeutung. Der Programmcode dieser Variablen kann sowohl unter WinPREP in der Methode *MALDI-LS 1.42s.mpt*, als auch in der MSL-Skriptdatei *MALDI-LS 1.42s.s* eingesehen werden. In dem dieser Arbeit getrennt beiliegenden Anhang sind Bereiche, in denen Definitionen der unterschiedlichen Laufzeitvariablen erfolgen grau hervorgehoben.

Tabelle 59: Volumina – Wichtige Laufzeitvariablen und deren Bedeutung im Programm *MALDI-LS 1.42s*

Laufzeitvariable	Position in der Methode (vergleiche Abbildung 74)	Bedeutung / Erklärung
Enzyme		
dDspVolume_S1	Verteilung und Thermostatisierung der Enzyme	Berechnet das für die Verteilung der Enzyme notwendige Gesamtvolumen in Abhängigkeit der Probenanzahl
dDspVolume_S2		
dDspVolume_S3		
Puffer		
dDspVolume_S4	Verteilung der Pufferlösungen	Berechnet das für die Verteilung der Puffer notwendige Gesamtvolumen in Abhängigkeit der Probenanzahl
dDspVolume_S5		
dDspVolume_S6		
dDspVolume_S7		
Matrices		
dDspVolume_S8	Verteilung der Pufferlösungen	Berechnet das für die Verteilung der Matrices notwendige Gesamtvolumen in Abhängigkeit der Probenanzahl
dDspVolume_S9		
Reagenz		
dDspVolume_S10	Verteilung der Reagenzlösung	Berechnet das für die Verteilung des Reagenzes notwendige Gesamtvolumen in Abhängigkeit der Probenanzahl

Tabelle 60: Zeiten – Wichtige Laufzeitvariablen und deren Bedeutung im Programm *MALDI-LS 1.42s*

Laufzeitvariable	Position in der Methode (vergleiche Abbildung 74)	Bedeutung / Erklärung
Proben		
stTimePeriod_S1	Timer für die Trocknung der Proben	Berechnet das für die Trocknung der auf das MALDI-Target aufgetragenen Proben nötige Zeitintervall
Matrices		
stTimePeriod_S2	Timer für die Matrixtrocknung	Berechnet das für die Trocknung der auf das MALDI-Target aufgetragenen Matrices nötige Zeitintervall

Alle Laufzeitvariablen zur Berechnung von Volumina beziehen sich auf Teil I von *MALDI-LS 1.42s* (Vorbereitung von Lösungen). Es wird dabei nach Berechnung nur soviel von den Stammlösungen entnommen, wie für die Bearbeitung der Proben notwendig ist.

Bei den Laufzeitvariablen, die einen Timer betreffen, ist insbesondere „*stTimePeriod_S1*“ wichtig. Durch Berechnung dieser Variable aus der Probenanzahl soll sichergestellt werden, dass die Probenlösung zunächst vollständig eingetrocknet ist, bevor weitere Lösungen aufge-

tragen werden. Dies betrifft insbesondere die nachfolgende Auftragung einer Enzymlösung, wenn direkt im Anschluss an die Auftragung der Proben ohne Derivatisierung sequenziert werden soll. Die Berechnung der Trocknungszeit gewährleistet, dass die weitere Abarbeitung so schnell wie möglich erfolgt, ohne dass das Enzym durch eventuell noch vorhandenes Lösungsmittel aus der Probenlösung in seiner Aktivität beeinflusst wird. Die empirisch optimierte Formel zur Berechnung der Trocknungszeit, wie sie für „*stTimePeriod_S1*“ eingegeben wurde, gilt natürlich nur für die getesteten Lösungsmittelkombinationen (wässrige Lösungen mit Acetonitril) und Spaltungstemperaturen (ca. $30\pm 3^\circ\text{C}$). Bei niedrigeren Spaltungstemperaturen oder anderen Lösungsmittelzusammensetzungen für die Proben muss ein entsprechender Faktor berücksichtigt werden. Auch im Bezug auf die Beibehaltung der Ortschäfe der Präparation auf einem bestimmten Probenspot müssen die Timer aller Reaktionsschritte so gewählt werden, dass erst nach dem Trocknen einer Lösung die nächste Lösung aufgetragen wird. Befindet sich durch zu schnelle Auftragung der folgenden Lösung zu einem bestimmten Zeitpunkt ein zu großes Volumen (etwa $>1,5\mu\text{l}$) auf einem MALDI-Spot, so breitet sich der Tropfen, je nach Acetonitrilgehalt, zu stark aus. Hierbei konnte in einigen Fällen eine Durchmischung benachbarter Spots beobachtet werden, insbesondere dann, wenn der Acetonitrilanteil einer Fraktion sehr hoch und damit die Oberflächenspannung des Tropfens sehr gering war.

3.1.4 Dateneingabe

Alle für den Ablaufs der Methode *MALDI-LS 1.42s* erforderlichen Daten werden über insgesamt vier Dateien eingelesen:

- II) FLAG PARAMETERS.TXT
- III) STOCK SOLUTION SOURCE.TXT
- IV) STOCK SOLUTION DESTINATION.TXT
- V) SAMPLES.TXT

Die Dateien liegen als Tabellen im Textformat vor. Die Tabellenspalten sind über Tabulatoren getrennt. Das Programm liest die einzelnen Informationen aus den jeweiligen Spalten aus. Die Eingabe der Informationen in eine solche Texttabelle wäre jedoch zu umständlich und unübersichtlich. Für die Eingabe wurde eine Vorlage mit dem Tabellenkalkulationsprogramm Microsoft Excel entworfen, die als simple Eingabemaske dient. Im folgenden soll eine kurze Beschreibung der vier Dateien gegeben werden. Dabei wird auch gezeigt, welche Bedeutung

die Eintragungen in den Tabellen für den Programmablauf der Methode *MALDI-LS 1.42s* besitzen.

3.1.4.1 Die Datei „FLAG PARAMETERS.TXT“

Die Datei „Flag Parameters“ enthält globale Angaben über die optionale Durchführung einzelner Schritte und über bestimmte Zeitparameter wie beispielsweise Reaktionszeiten der Derivatisierung oder Sequenzierung. Tabelle 61 zeigt die einzutragenden Schalter.

Tabelle 61: Die Datei „FLAG PARAMETERS.TXT“

Spalte Nr.	Eintrag	Schalter ¹	Bedeutung
1	Multiple Matrix Addition	0 / 1	entscheidet darüber, ob eine doppelte Auftragung der Matrix erfolgt: 0=nein, 1=ja
2	Use Enzyme #1	E1	Entscheidet über die Ausführung des Verteilungsschritts für Enzym #1
3	Use Enzyme #2	E2	Entscheidet über die Ausführung des Verteilungsschritts für Enzym #2
4	Use Enzyme #3	E3	Entscheidet über die Ausführung des Verteilungsschritts für Enzym #3
5	Use DHB	DHB	Entscheidet über die Ausführung des Verteilungsschritts für die DHB-Matrix
6	Use alpha	alpha	Entscheidet über die Ausführung des Verteilungsschritts für die CHCA-Matrix
7	Use Buffer #1	B1	Entscheidet über die Ausführung des Verteilungsschritts für Puffer #1
8	Use Buffer #2	B2	Entscheidet über die Ausführung des Verteilungsschritts für Puffer #2
9	Use Buffer #3	B3	Entscheidet über die Ausführung des Verteilungsschritts für Puffer #3
10	Use Buffer #4	B4	Entscheidet über die Ausführung des Verteilungsschritts für Puffer #4
11	Use Reagent	R1	Entscheidet über die Ausführung des Verteilungsschritts für das Derivatisierungsreagenz
12	Buffer #1 Time	d hh:mm:ss	Trocknungszeit für Puffer #1
13	Buffer #2 Time	d hh:mm:ss	Trocknungszeit für Puffer #2
14	Reaction Time	d hh:mm:ss	Reaktionszeit
15	Digest Time Enzyme #1	d hh:mm:ss	Verdauzeit mit Enzym #1
16	Digest Time Enzyme #2	d hh:mm:ss	Verdauzeit mit Enzym #2
17	Digest Time Enzyme #3	d hh:mm:ss	Verdauzeit mit Enzym #3
18	Calibrant	C1	Entscheidet über die Auftragung des Kalibranden
19	FTP-Transfer	FTP	Entscheidet über den Datentransfer via FTP zum Steuerungsrechner des MALDI-MS
20	Enzyme Preincubation	d hh:mm:ss	Vorinkubationszeit der Enzyme im Heizblock von „Eppendorf 0.5 HUE“

¹ Angabe d hh:mm:ss entspricht einer Zeitangabe in Tagen (d), Stunden (h), Minuten (m) und Sekunden (s)

Die Schalter der Spalten Nummern 1 bis 11, 18 und 19 entscheiden über die optionale Ausführung einzelner Schritte in der Methode *MALDI-LS 1.42s*. Sind einzelne Schalter in den Spalten 2 bis 11 in der Datei eingetragen, so wird der Schritt in der Methode durchgeführt, im anderen Fall wird der Schritt ausgelassen. Ist für den Eintrag „Multiple Matrix Addition“ der Schalter auf 1 (= ja) gesetzt, so wird das Matrixvolumen im entsprechenden Verteilungsschritt angepasst (verdoppelt). Diese Kategorie der Flagparameter nimmt also im wesentlichen Einfluss darauf, welche Schritte bei der Vorbereitung der Lösungen in Teil I ausgeführt werden müssen und welche Schritte ausgelassen werden können. Neben diesen Parametern, die eine reine „ja/nein-Schalterfunktion“ besitzen, sind in der Datei „FLAG PARAMETERS.TXT“ in den Spalten 12 bis 17 und 20 definierte Zeitangaben für einige Timer zu machen. Diese betreffen vor allen die gewünschte, minimale Spaltungsdauer mit den einzelnen Exopeptidasen. Der nächste Schritt in einer Methode wird frühestens dann ausgeführt, wenn die für den Timer eingetragene Zeit abgelaufen ist.

3.1.4.2 Die Dateien „STOCK SOLUTION SOURCE.TXT“ und „STOCK SOLUTION DESTINATION.TXT“

Über die Dateien „STOCK SOLUTION SOURCE.TXT“ und „STOCK SOLUTION DESTINATION.TXT“ werden die Aufnahme- und Abgabepositionen (Elementbezeichnung = Rack Name + Position) für die Verteilungsschritte von Enzym-, Matrix-, Puffer- und Reagenzlösungen festgelegt.

Tabelle 62: Die Datei „STOCK SOLUTION SOURCE.TXT“

Spalte	Eintrag	Standardwert	Bedeutung
1	SS-Enzyme Plate I.D.	Eppendorf 0.5 EMC	Rack Name für Stammlösungen der Enzyme
2	SS-Enzyme #1 Position	17	Position der Stammlösung von Enzym #1, #2 oder #3 auf dem betreffenden Rack
3	SS-Enzyme #2 Position	18	
4	SS-Enzyme #3 Position	19	
9	SS-DHB-Matrix Plate I.D.	Eppendorf 1.5 SSMBR	Rack Name für die Stammlösung der DHB-Matrix
6	SS-DHB-Position	1	Position der DHB-Stammlösung auf dem betreffenden Rack
7	SS-alpha-Matrix Plate I.D.	Eppendorf 1.5 SSMBR	Rack Name für die Stammlösung der CHCA-Matrix
8	SS-alpha Position	2	Position der CHCA-Stammlösung auf dem betreffenden Rack

Spalte	Eintrag	Standardwert	Bedeutung
9	SS-Buffer Plate I.D.	Eppendorf 1.5 SSMBR	Rack Name für Stammlösungen der Puffer #1, #2, #3 und #4
10	SS-Buffer #1 Position	5	Position der Stammlösungen von Puffer #1 oder #2 auf dem betreffenden Rack
11	SS-Buffer #2 Position	6	
12	SS-Reagent Plate I.D.	Eppendorf 1.5 SSMBR	Rack Name für die Stammlösungen des Derivatisierungsreagenz
13	SS-Reagent Position	9	Position der Stammlösung von Enzym #1 auf dem betreffenden Rack
14	SS-Buffer #3 Position	7	Position der Stammlösungen von Puffer #3 oder #4 auf dem betreffenden Rack
15	SS-Buffer #4 Position	8	

Tabelle 63: Die Datei „STOCK SOLUTION DESTINATION.TXT“

Spalte	Eintrag	Standardwert	Bedeutung
1	Enzyme Plate I.D.	Eppendorf 0.5 HUE	Rack Name für die Enzymlösungen
2	Enzyme #1 Position	1 – 4	Zielpositionen der Lösung von Enzym #1, #2 und #3 auf dem betreffenden Rack
3	Enzyme #2 Position	5 – 8	
4	Enzyme #3 Position	9 – 12	
9	DHB-Matrix Plate I.D.	Eppendorf 0.5 EMC	Rack Name für die DHB-Lösungen
6	DHB-Position	1 – 4	Zielposition der DHB-Lösung auf dem betreffenden Rack
7	alpha-Matrix Plate I.D.	Eppendorf 0.5 EMC	Rack Name für die CHCA-Lösungen
8	alpha Position	5 – 8	Zielposition der CHCA-Lösung auf dem betreffenden Rack
9	Buffer #1 Plate I.D.	Eppendorf 0.5 EMC	Rack Name für die Lösung von Puffer #1
10	Buffer #1 Position	9 – 12	Zielposition der Lösung von Puffer #1 auf dem betreffenden Rack
11	Buffer #2 Plate I.D.	Eppendorf 0.5 EMC	Rack Name für die Lösung von Puffer #2
12	Buffer #2 Position	13 – 16	Zielposition der Lösung von Puffer #2 auf dem betreffenden Rack
13	Reagent Plate I.D.	Eppendorf 1.5 SSMBR	Rack Name für die Lösung des Reagenz
14	Reagent Position	13 – 16	Zielposition der Reagenzlösung auf dem betreffenden Rack
15	Buffer #3 Plate I.D.	Eppendorf 1.5 SSMBR	Rack Name für die Lösung von Puffer #3
16	Buffer #3 Position	17 – 20	Zielposition der Lösung von Puffer #3 auf dem betreffenden Rack
17	Buffer #4 Plate I.D.	Eppendorf 1.5 SSMBR	Rack Name für die Lösung von Puffer #4
18	Buffer #4 Position	21 – 24	Zielposition der Lösung von Puffer #4 auf dem betreffenden Rack

3.1.4.3 Die Datei „SAMPLES.TXT“ – Gemeinsame Dateneingabe für MultiPROBE II und AutoXecute

Der größte und wichtigste Teil der Parameter wird dem Programm durch die Datei „SAMPLES.TXT“ übergeben. Informationen über Entnahmepositionen von Puffer-, Reagenz-, Enzym- oder Matrixlösung sind bereits über Datei „STOCK SOLUTION DESTINATION.TXT“ definiert worden. Diese Informationen tauchen daher in der Datei „SAMPLES.TXT“ nicht mehr explizit auf. Angaben über die Aufnahmepositionen der Proben und die Abgabepositionen der Lösungen auf dem MALDI-Target werden dagegen in der Datei „SAMPLES.TXT“ festgelegt. Da sich die Art und Anzahl der Probenracks und MALDI-Targets nach der Anzahl der Proben richtet, erfolgt eine Definition dieser Elemente für jedes Problem individuell in der Datei „SAMPLES.TXT“ und nicht in der Methode direkt. Für „Sample Plate I.D.“ und „Sample Position“ sowie für „MALDI Plate I.D.“ und „Pos_on_SCOUT“ müssen die betreffenden SRTING-Werte der Racks und INTEGER-Werte der Proben- und Zielpositionen eingetragen werden.

Tabelle 64: Die Datei „SAMPLES.TXT“

Spalte	Eintrag	Schalter / Standardwert ¹	Bedeutung
1	Comment_1		Allgemeine Informationen über die Probe
2	Sample Plate I.D.	STRING	Rack Name für Proben / Kalibranden
3	Sample Position	INTEGER	Aufnahmenpositionen der Proben auf dem jeweiligen Rack
4	Sample Volume	FLOAT	Aufnahmevolumen der Probe
9	Buffer #1 Volume	FLOAT	Aufnahmevolumen von Puffer #1
6	Buffer #1 Flag	B1	Entscheidet über die Auftragung von Puffer #1 auf die jeweilige Probe
7	Buffer #2 Volume	FLOAT	Aufnahmevolumen von Puffer #2
8	Buffer #2 Flag	B2	Entscheidet über die Auftragung von Puffer #2 auf die jeweilige Probe
9	Reagent Volume	FLOAT	Aufnahmevolumen der Reagenzlösung
10	Reagent Flag	R1	Entscheidet über die Auftragung von Reagenz auf die jeweilige Probe
11	Enzyme #1 Volume	FLOAT	Aufnahmevolumen von Enzym #1
12	Enzyme #1 Flag	E1	Entscheidet über die Auftragung von Enzym #1 auf die jeweilige Probe
13	Enzyme #2 Volume	FLOAT	Aufnahmevolumen von Enzym #2
14	Enzyme #2 Flag	E2	Entscheidet über die Auftragung von Enzym #2 auf die jeweilige Probe
15	Enzyme #3 Volume	FLOAT	Aufnahmevolumen von Enzym #3

Spalte	Eintrag	Schalter / Standardwert ¹	Bedeutung
16	Enzyme #3 Flag	E3	Entscheidet über die Auftragung von Enzym #3 auf die jeweilige Probe
17	Matrix Volume	<i>FLOAT</i>	Aufnahmevolumen für die erste Matrixaufnahme
18	Matrix Volume optional	<i>FLOAT</i>	Aufnahmevolumen für die zweite Matrixaufnahme
19	Matrix optional Flag	DHB_o / alpha_o	Entscheidet über die zweite (optionale) Auftragung von DHB- oder CHCA-Matrix auf die jeweilige Probe
20	Matrix Flag	DHB / alpha	Entscheidet über die Auftragung von DHB- oder CHCA-Matrix auf die jeweilige Probe
21	Preparation	-	Bezeichnung der Matrixpräparation
22	Buffer #3 Volume	<i>FLOAT</i>	Aufnahmevolumen von Puffer #3
23	Buffer #3 Flag	B3	Entscheidet über die Auftragung von Puffer #3 auf die jeweilige Probe
24	Buffer #4 Volume	<i>FLOAT</i>	Aufnahmevolumen von Puffer #4
25	Buffer #4 Flag	B4	Entscheidet über die Auftragung von Puffer #4 auf die jeweilige Probe
26	MALDI Plate I.D.	<i>STRING</i>	Rack Name des MALDI-Targets
27	Pos_on_SCOUT	<i>INTEGER</i>	Zielposition der Probe auf dem MALDI-Target
28	AutoX_Method	<i>STRING</i>	Dateiname des ausführenden Skripts für die automatische MALDI-MS Messung mittels AutoXecute
29	Spectrum_Filename	<i>STRING</i>	gewünschter Dateiname des aufgenommenen MALDI-Massenspektrums
30	Spectrum_Directory	<i>STRING</i>	gewünschtes Ablageverzeichnis der Datei des aufgenommenen MALDI-Massenspektrums
31	Probe_Geometry	SCOUT26	für die verwendete Target-Geometrie bindender Eintrag – in dieser Arbeit SCOUT 26

¹ STRING=Zeichenkette, INTEGER=Ganzzahl; FLOAT=Fließkommazahl;

Der Vorteil einer Eingabe der Daten über die Tabelle ist offensichtlich: für jede einzelne Probe kann individuell festgelegt werden, welche Schritte zu bearbeiten sind und welche Volumina aufgetragen werden sollen. Bei einer ebenfalls möglichen Direkteingabe dieser Informationen über die grafische Oberfläche der WinPREP-Software müsste die Methode unter Umständen für jede Sequenzierung umgearbeitet werden. Dabei wären die Fehleranfälligkeit und der Zeitaufwand bei der Eingabe sicherlich wesentlich größer als bei einer Dateneingabe über eine Tabelle.

Ob ein Schritt in der Methode für eine bestimmte Probe ausgeführt werden soll, wird in der Datei „SAMPLES.TXT“, über sogenannte Schalter festgelegt. Sind die Schalter B1, B2, B3, B4, R1, E1, E2 oder E3 gesetzt, so wird der Pipettierschritt durchgeführt. Bei der Matrix-

lösung erfolgt je nach Schalter entweder die Auftragung von DHB (Schalter: DHB) oder CHCA (Schalter: alpha). Für eine optionale, zweite Auftragung der Matrix gilt Analoges (DHB_o oder alpha_o). Soll ein Pipettierschritt ausgeführt werden, so muss zudem ein Volumen angegeben werden. Eine notwendige Volumeneingabe ist in Tabelle 64 über den Eintrag „*FLOAT*“ kenntlich gemacht. Hier ist eine Volumenangabe in Form einer Fließkommazahl zu machen, wobei alle Volumeneinträge in μl vorzunehmen sind.

Die Spalten 1, sowie 27 bis 31 enthalten Angaben, die entweder der gemeinsamen Nutzung durch den MultiPROBE II und die AutoXecute des MALDI-MS dienen (schwarze Felder) oder ausschließlich durch die AutoXecute des MALDI-MS verarbeitet werden (graue Felder). Die gemeinsame Eingabe von Daten in eine Tabelle ist möglich, da analog zur MSL Programmsprache (MultiPROBE II) auch die Automatik-Software AutoXecute 4.2 des MALDI-MS die benötigten Daten aus einer über Tabulatoren definierten Tabelle im Textformat ausgelesen kann. In die Spalte „Comment_1“ können beliebige Informationen über die Probe eingetragen werden. Diese Informationen geben einerseits einen einfacheren Überblick über die Probentabelle „SAMPLES.TXT“, andererseits tauchen die hier gemachten Angaben später beim automatischen Ausdruck des MALDI-MS-Spektrums wieder auf. Somit kann eine eindeutige Zuordnung der letztendlich erhaltenen Spektren zu den zugehörigen Proben erfolgen. Ebenso gemeinsam genutzt von MultiPROBE und AutoXecute sind die Informationen in der Spalte „Pos_on_SCOUT“. Während hier für den MultiPROBE II die Zielpositionen der Flüssigkeitsabgaben festgelegt sind, definieren diese Werte Zeile für Zeile die Reihenfolge der Abarbeitung der Probenspots bei der automatischen MALDI-MS Messung durch die AutoXecute.

Die Inhalte der Spalten 28 bis 31 in „SAMPLES.TXT“ definieren Parameter, die später exklusiv von der MALDI-Software AutoXecute benötigt werden. Eine detaillierte Beschreibung der Parameter erfolgt unter Punkt 3.2 (Automatische MALDI-MS Messungen) im nächsten Kapitel.

Beispiele für die vier oben beschriebenen Dateien FLAG PARAMETERS, STOCK SOLUTION SOURCE, STOCK SOLUTION DESTINATION und SAMPLES sind auf der folgenden Doppelseite dargestellt. Gezeigt sind die für die Dateneingabe genutzten Microsoft Excel-Formate der Dateien (XLS-Formate). Damit diese Eingabedateien durch MultiPROBE- und MALDI-MS-System genutzt werden können, werden die Dateien in einem Tabulator getrennten Textformat abgespeichert. Am Beispiel der Datei SAMPLES.XLS ist die Flexibilität der Methode aufgezeigt, die es gestattet, jede Probe mit weitgehend individuellen Parametern zu sequenzieren (Erklärungen siehe folgende Doppelseite).

● **Beispiel für die Eingabeformat der Datei "FLAG PARAMETERS.TXT" in Form einer Microsoft Excel-Tabelle (Datei: "FLAG PARAMETERS.XLS")**

Multiple Matrix Addition	Use Enzyme #1	Use Enzyme #2	Use Enzyme #3	Use DHB	Use alpha	Use Buffer #1	Use Buffer #2	Use Buffer #3	Use Buffer #4	Use Reagent	Buffer #1 Time	Buffer #2 Time	Reaction Time	Digest Time Enzyme #1	Digest Time Enzyme #2	Digest Time Enzyme #3	Calibrant	FTP	Enzyme Pre-Incubation
0 / 1	E1	E2	E3	DHB	alpha	B1	B2	B3	B4	R1	d hh:mm:ss	d hh:mm:ss	d hh:mm:ss	d hh:mm:ss	d hh:mm:ss	d hh:mm:ss	C1	FTP	d hh:mm:ss

● **Beispiel für die Eingabeformat der Datei "STOCK SOLUTION SOURCE.TXT" in Form einer Microsoft Excel-Tabelle (Datei: "STOCK SOLUTION SOURCE.XLS")**

SS-Enzyme Plate I.D.	SS-Enzyme 1 Position	SS-Enzyme 2 Position	SS-Enzyme 3 Position	SS-DHB-Matrix Plate I.D.	SS-DHB Position	SS-alpha-Matrix Plate I.D.	SS-alpha Position	SS-Buffer Plate I.D.	SS-Buffer #1 Position	SS-Buffer #2 Position	SS-Reagent Plate I.D.	SS-Reagent Position	SS-Buffer #3 Position	SS-Buffer #4 Position
Eppendorf 0.5 EMC	17	18	19	Eppendorf 1.5 SSMBR	1	Eppendorf 1.5 SSMBR	2	Eppendorf 1.5 SSMBR	5	6	Eppendorf 1.5 SSMBR	9	7	8

● **Beispiel für die Eingabeformat der Datei "STOCK SOLUTION DESTINATION.TXT" in Form einer Microsoft Excel-Tabelle (Datei: "STOCK DESTINATION SOURCE.XLS")**

Enzyme Plate I.D.	Enzyme 1 Position	Enzyme 2 Position	Enzyme 3 Position	DHB-Matrix Plate I.D.	DHB Position	alpha-Matrix Plate I.D.	alpha Position	Buffer #1 Plate I.D.	Buffer Position	Buffer #2 Plate I.D.	Buffer Position	SS-Reagent Plate I.D.	Reagent Position	Buffer #3 Plate I.D.	Buffer Position	Buffer #4 Plate I.D.	Buffer Position
Eppendorf 0.5 HUE	1	5	9	Eppendorf 0.5 EMC	1	Eppendorf 0.5 EMC	5	Eppendorf 0.5 EMC	9	Eppendorf 0.5 EMC	13	Eppendorf 1.5 SSMBR	13	Eppendorf 1.5 SSMBR	17	Eppendorf 1.5 SSMBR	21
Eppendorf 0.5 HUE	2	6	10	Eppendorf 0.5 EMC	2	Eppendorf 0.5 EMC	6	Eppendorf 0.5 EMC	10	Eppendorf 0.5 EMC	14	Eppendorf 1.5 SSMBR	14	Eppendorf 1.5 SSMBR	18	Eppendorf 1.5 SSMBR	22
Eppendorf 0.5 HUE	3	7	11	Eppendorf 0.5 EMC	3	Eppendorf 0.5 EMC	7	Eppendorf 0.5 EMC	11	Eppendorf 0.5 EMC	15	Eppendorf 1.5 SSMBR	15	Eppendorf 1.5 SSMBR	19	Eppendorf 1.5 SSMBR	23
Eppendorf 0.5 HUE	4	8	12	Eppendorf 0.5 EMC	4	Eppendorf 0.5 EMC	8	Eppendorf 0.5 EMC	12	Eppendorf 0.5 EMC	16	Eppendorf 1.5 SSMBR	16	Eppendorf 1.5 SSMBR	20	Eppendorf 1.5 SSMBR	24

● **Beispiel für die Eingabeformat der Datei "STOCK SOLUTION DESTINATION.TXT" in Form einer Microsoft Excel-Tabelle (Datei: "STOCK DESTINATION SOURCE.XLS")**

Beispiel für die Möglichkeit der individuellen Bearbeitung (Sequenzierung) einzelner Proben:

- Die HPLC-Fractionen 1-6 werden mit Enzym #1 sequenziert. Um die Spaltungszeit zu verlängern wird nach einer festgelegten Zeitspanne (in Datei "FLAG PARAMETERS.TXT"; siehe Text) 1.0µl von Puffer #3 zugegeben
- Die HPLC-Fractionen 7-12 werden mit Enzym #2 sequenziert. In diesem Fall soll keine Verlängerung der Spaltungszeit erfolgen.
- Die HPLC-Fractionen 13-18 werden mit einer sequentiellen Kombination der Enzyme #1 und #2 behandelt. Dabei erfolgt nach Spaltung mit Enzym #1 vor der Auftragung von Enzym #2 eine Anpassung der pH-Bedingungen über die Auftragung von 1,0µl von Puffer #4
- Die HPLC-Fractionen 19-24 werden mit Enzym #3 sequenziert, wobei hier abschließend eine doppelte Auftragung der DHB-Matrix erfolgt (z.B. weil aufgrund eines bestimmten Puffersalzes in Enzym #3 erst nach doppelter Matrixauftragung die Matrixkristalle gut ausgebildet werden)

Comment_1	Sample Plate I.D.	Sample Position	Sample Volume	Buffer #1 Volume	Buffer #1 Flag	Buffer #2 Volume	Buffer #2 Flag	Reagent Volume	Reagent Flag	Enzyme #1 Volume	Enzyme #1 Flag	Enzyme #2 Volume	Enzyme #2 Flag	Enzyme #3 Volume	Enzyme #3 Flag	Matrix Volume	Matrix optional Volume	Matrix optional Flag	Matrix Flag	Preparation	Buffer #3 Volume	Buffer #3 Flag	Buffer #4 Volume	Buffer #4 Flag	MALDI Plate I.D.	Pos_on_S cout	AutoX_Method	Spectrum_Filename	Spectrum_Directory	Probe_Geometry
Calibrant (Ang2 / Bra / SubP / Neu / ACTH); 1µl; DHB(x2)	Eppendorf 0.5 EMC	21	0,5																DHB	DD (50/50)					MALDI1	26	multiprobe_cal_5	000308-MYG_HORSE-Cal	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 1	Eppendorf 0.5 Samples 1	1	0,5							1,0	E1					1,0			DHB	DD (50/50)	1,0	B3			MALDI1	1	multiprobe_5	000308-MYG_HORSE-F1	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 2	Eppendorf 0.5 Samples 1	2	0,5							1,0	E1					1,0			DHB	DD (50/50)	1,0	B3			MALDI1	2	multiprobe_5	000308-MYG_HORSE-F2	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 3	Eppendorf 0.5 Samples 1	3	0,5							1,0	E1					1,0			DHB	DD (50/50)	1,0	B3			MALDI1	3	multiprobe_5	000308-MYG_HORSE-F3	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 4	Eppendorf 0.5 Samples 1	4	0,5							1,0	E1					1,0			DHB	DD (50/50)	1,0	B3			MALDI1	4	multiprobe_5	000308-MYG_HORSE-F4	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 5	Eppendorf 0.5 Samples 1	5	0,5							1,0	E1					1,0			DHB	DD (50/50)	1,0	B3			MALDI1	5	multiprobe_5	000308-MYG_HORSE-F5	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 6	Eppendorf 0.5 Samples 1	6	0,5							1,0	E1					1,0			DHB	DD (50/50)	1,0	B3			MALDI1	6	multiprobe_5	000308-MYG_HORSE-F6	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 7	Eppendorf 0.5 Samples 1	7	0,5									1,0	E2			1,0			DHB	DD (50/50)					MALDI1	7	multiprobe_5	000308-MYG_HORSE-F7	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 8	Eppendorf 0.5 Samples 1	8	0,5									1,0	E2			1,0			DHB	DD (50/50)					MALDI1	8	multiprobe_5	000308-MYG_HORSE-F8	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 9	Eppendorf 0.5 Samples 1	9	0,5									1,0	E2			1,0			DHB	DD (50/50)					MALDI1	9	multiprobe_5	000308-MYG_HORSE-F9	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 10	Eppendorf 0.5 Samples 1	10	0,5									1,0	E2			1,0			DHB	DD (50/50)					MALDI1	10	multiprobe_5	000308-MYG_HORSE-F10	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 11	Eppendorf 0.5 Samples 1	11	0,5									1,0	E2			1,0			DHB	DD (50/50)					MALDI1	11	multiprobe_5	000308-MYG_HORSE-F11	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 12	Eppendorf 0.5 Samples 1	12	0,5									1,0	E2			1,0			DHB	DD (50/50)					MALDI1	12	multiprobe_5	000308-MYG_HORSE-F12	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 13	Eppendorf 0.5 Samples 1	13	0,5							1,0	E1	1,0	E2			1,0			DHB	DD (50/50)			1,0	B4	MALDI1	13	multiprobe_5	000308-MYG_HORSE-F13	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 14	Eppendorf 0.5 Samples 1	14	0,5							1,0	E1	1,0	E2			1,0			DHB	DD (50/50)			1,0	B4	MALDI1	14	multiprobe_5	000308-MYG_HORSE-F14	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 15	Eppendorf 0.5 Samples 1	15	0,5							1,0	E1	1,0	E2			1,0			DHB	DD (50/50)			1,0	B4	MALDI1	15	multiprobe_5	000308-MYG_HORSE-F15	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 16	Eppendorf 0.5 Samples 1	16	0,5							1,0	E1	1,0	E2			1,0			DHB	DD (50/50)			1,0	B4	MALDI1	16	multiprobe_5	000308-MYG_HORSE-F16	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 17	Eppendorf 0.5 Samples 1	17	0,5							1,0	E1	1,0	E2			1,0			DHB	DD (50/50)			1,0	B4	MALDI1	17	multiprobe_5	000308-MYG_HORSE-F17	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 18	Eppendorf 0.5 Samples 1	18	0,5							1,0	E1	1,0	E2			1,0			DHB	DD (50/50)			1,0	B4	MALDI1	18	multiprobe_5	000308-MYG_HORSE-F18	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 19	Eppendorf 0.5 Samples 1	19	0,5											1,0	E3	1,0	1,0	DHB_o	DHB	DD (50/50)					MALDI1	19	multiprobe_5	000308-MYG_HORSE-F19	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 20	Eppendorf 0.5 Samples 1	20	0,5											1,0	E3	1,0	1,0	DHB_o	DHB	DD (50/50)					MALDI1	20	multiprobe_5	000308-MYG_HORSE-F20	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 21	Eppendorf 0.5 Samples 1	21	0,5											1,0	E3	1,0	1,0	DHB_o	DHB	DD (50/50)					MALDI1	21	multiprobe_5	000308-MYG_HORSE-F21	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 22	Eppendorf 0.5 Samples 1	22	0,5											1,0	E3	1,0	1,0	DHB_o	DHB	DD (50/50)					MALDI1	22	multiprobe_5	000308-MYG_HORSE-F22	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 23	Eppendorf 0.5 Samples 1	23	0,5											1,0	E3	1,0	1,0	DHB_o	DHB	DD (50/50)					MALDI1	23	multiprobe_5	000308-MYG_HORSE-F23	/data/Robert/AutoXecute	SCOUT26
MYG_HORSE; P02188 (Trypsinspaltung); HPLC Fraktion 24	Eppendorf 0.5 Samples 1	24	0,5											1,0	E3	1,0	1,0	DHB_o	DHB	DD (50/50)					MALDI1	24	multiprobe_5	000308-MYG_HORSE-F24	/data/Robert/AutoXecute	SCOUT26

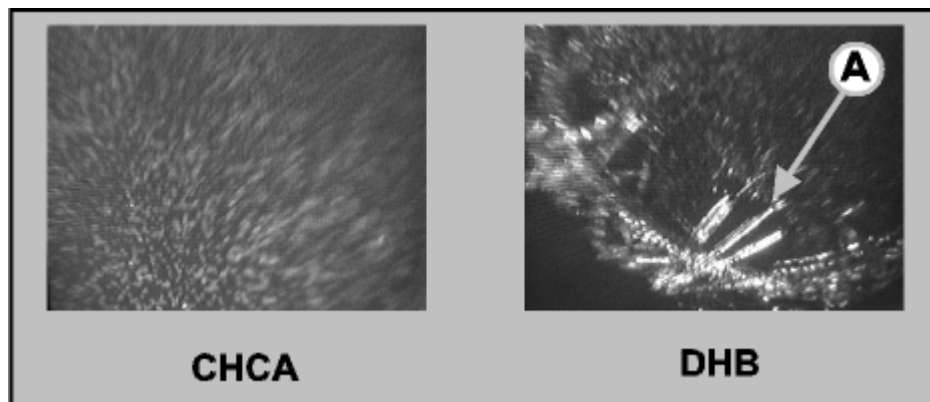
3.1.4.4 Datenaustausch zwischen MultiPROBE II und MALDI-MS

Die integrierte Eingabe der Daten in „SAMPLES.TXT“ für die Probenpräparation mittels MultiPROBE II und automatische MALDI-MS Messung mittels AutoXecute erfordert eine einfache Weiterleitung der Daten vom Ort der Probeneingabe – hier dem Steuerungsrechner für MultiPROBE II – zum Steuerungsrechner des MALDI-MS mit der AutoXecute Software. Der Datentransport erfolgt im Rahmen der Methode *MALDI-LS 1.42s* via FTP über ein in die Methode integriertes, ausführbares Programm. Die Aufgabe dieses FTP-Programms ist es, die aktuell zu bearbeitende Parameterdatei „SAMPLES.TXT“ auf dem MALDI-MS Steuerungsrechner zu kopieren und dort für die weitere Benutzung durch die AutoXecute Software direkt verfügbar zu machen. In der vorliegenden Arbeit wurde dieses Programm mit den Scripting Editor der Software „FTP Control“ geschrieben und mit dem zugehörigen Compiler in das ausführbare Programm übersetzt. Dieses Programm, welches zusammen mit der Methode *MALDI-LS 1.42s* arbeitet heißt „*MALDI2.EXE*“. Der Quellcode in der Datei *MALDI2.FTP* liegt in gedruckter Form dem getrennten Anhang und als Textdatei auf CD-ROM dieser Arbeit bei (Anhang E). Angaben bezüglich Verzeichnispfaden, Host- und Usernamen in *MALDI2.FTP* und *MALDI2.EXE* sind natürlich spezifisch für die in dieser Arbeit verwendete Konfiguration.

3.2 Automatische MALDI-MS Messungen

Das Prinzip der automatischen MALDI-MS Messung basiert auf einer rasterförmigen Abtastung der einzelnen Probenspots auf dem Target mit den UV-Desorptionslaser. Dabei werden an verschiedenen Stellen der Präparation Spektren unter Berücksichtigung vorgegebener Gütekriterien, wie z.B. Auflösung oder Signal-Rausch-Verhältnis, aufgenommen, summiert und schließlich abgespeichert. Die Kontrolle über die Messung erfolgt hierbei über eine *Fuzzy Control* Einheit der AutoXecute Software. Die *Fuzzy Control* überprüft die Peakintensität und Auflösung der Spektren aus XACQ (Primärakquisition), vergleicht diese mit den geforderten Qualitätskriterien, passt falls notwendig die Laserabschwächung entsprechend an und summiert letztlich die Spektren aus der XACQ. Um im weiteren die einzelnen Einstellungen in der erarbeiteten AutoXecute-Methode für die Messung der sequenzierten Proben zu verstehen muss zunächst kurz auf die Besonderheiten der DHB-Präparationen eingegangen werden. Abbildung 77 zeigt einen Blick auf die morphologische Beschaffenheit von Probenpräparationen mit CHCA- und DHB-Matrix. Die Aufnahmen wurden über die Beobachtungsoptik (CCD-Kamera) des MALDI-MS erhalten (Randunschärfen in den Bildern von Abbildung 77 und folgenden sind mit der gegebenen Optik bei dieser Vergrößerung technisch bedingt und nicht vermeidbar).

Abbildung 77: Beschaffenheit der Matrixpräparation von CHCA- und DHB-Matrices



Deutlich zu sehen ist der Unterschied in der Homogenität der beiden Präparationen. Während die CHCA-Präparation eine sehr gleichmäßige Verteilung kleiner Matrixkristalle zeigt und in dieser Vergrößerung noch fast amorph wirkt, weist die DHB-Präparation ein äußerst unregelmäßiges Erscheinungsbild auf. Einzelne Kristalle, z.B. bei (A), sind in der DHB-Präparation sehr ausgeprägt, während in anderen Bereichen des Spots kaum Matrixkristalle zu finden sind. Zusätzlich befinden sich die Kristalle im allgemeinen auf den begrenzenden Rändern der Target-Spots. Diese Ränder weisen eine Vertiefung von wenigen zehntel Millimetern auf. Die leicht raue Beschaffenheit der Ränder scheint die Entstehung von Kristallen in dieser Spotregion zu begünstigen. Messungen an verschiedenen Stellen einer CHCA-Präparation zeigen zumeist, wie dies auch aufgrund des homogenen Aussehens zu erwarten ist, relativ ähnliche Spektren im Bezug auf die beobachteten Peaks und deren Intensität. Die Messungen an verschiedenen Stellen einer DHB-Präparation dagegen zeigen zum Teil große Unterschiede in der Intensität einzelner Peptide. Die besten Spektren erhält man in der Regel beim Beschuss der gut ausgebildeten Kristalle, wie z.B. bei (A). Je besser die Kristalle der Präparation dabei ausgebildet sind, desto konzentriert liegt die Probe meist in diesen Kristallen vor. Orte einer solchen Präparation, die beim Laserbeschuss eine sehr hohe Signalintensität liefern, werden als sogenannte *hot spots* bezeichnet.

Die beschriebenen Eigenschaften und Unterschiede der beiden Matrix-Präparationen haben zur Folge, dass DHB-Präparationen gegenüber CHCA-Präparationen für automatische Messungen zunächst ein wesentlich größeres Problem darstellen. Während sich beim Messen der CHCA-Präparation an nahezu jeder beschossenen Stelle ein Spektrum findet, kann die automatische Suche nach einem guten Spot oder einem *hot spot* bei der DHB-Präparation sehr langwierig sein.

Im Hinblick auf Probleme bei der automatischen Messung von DHB-Präparationen mussten deshalb zunächst die Parameter der rasterförmigen Spotabtastung und der eigentlichen Meßmethode in der AutoXecute optimiert werden. Zudem wurde im Zusammenhang mit der Morphologie der Matrix auch untersucht, ob sich durch veränderte Wahl des Lösungsmittels für die Matrix auch bei der DHB-Präparation eine homogenere Kristallisation erreichen lässt.

3.2.1.1 Optimierung der Probenpräparation

Durch Variation des Lösungsmittels für die DHB-Matrix wurde versucht eine homogenere Matrixschicht, ähnlich der CHCA-Präparation zu erzeugen. Die gezeigten Fotografien in Abbildung 77 basieren auf den erwähnten *dried-droplet* Standardpräparationen mit CHCA in Acetonitril/Wasser 50/50 (v/v) + 0,1% TFA und DHB in Acetonitril/Wasser 30/70 (v/v) + 0,1% TFA. In Vorversuchen zur Automatisierung wurden die MALDI-MS Spektren von mehreren synthetische Peptiden nach DHB-Präparationen verglichen, wobei folgende Lösungsmittel für die DHB-Matrix getestet wurden:

- ❖ Aceton + 0,1% TFA
- ❖ Acetonitril/Wasser 30/70 (v/v) + 0,1% TFA
- ❖ Acetonitril/Wasser 50/50 (v/v) + 0,1% TFA
- ❖ Acetonitril/Wasser 70/30 (v/v) + 0,1% TFA
- ❖ Acetonitril/Wasser 90/10 (v/v) + 0,1% TFA

Der Versuch einer Dünnschichtpräparation mit DHB in Aceton analog der bekannten Dünnschichtpräparation mit CHCA zeigte zwar eine gleichmäßige, feinkristalline Matrixschicht, jedoch waren die hieraus erhaltenen MALDI-MS Spektren bezüglich Intensität und Auflösung sehr schlecht. Hinzu kommen zwei weitere Nachteile des Acetons, die dieses Lösungsmittel insbesondere für eine direkte, unverdünnte Auftragung auf das MALDI-Target ungeeignet erscheinen lassen. Zum einen lassen sich Lösungen in 100% Aceton aufgrund der geringen Oberflächenspannung und des hohen Dampfdrucks sehr schlecht handhaben. Zum anderen „zerfließt“ der Acetontropfen beim Auftragen auf das Target sehr stark und eine ortsgenaue Präparation ist somit nicht möglich.

Bei Verwendung unterschiedlicher Acetonitrilgehalte waren die Spektren der Peptide bezüglich Intensität und Auflösung bis 70% Acetonitril im Lösungsmittel für die DHB-Matrix vergleichbar. Je höher der Acetonitrilgehalt, desto gleichmäßiger und feinkristalliner erscheint die beobachtete Morphologie der Präparation. Allerdings zeigen sich auch hier weiterhin

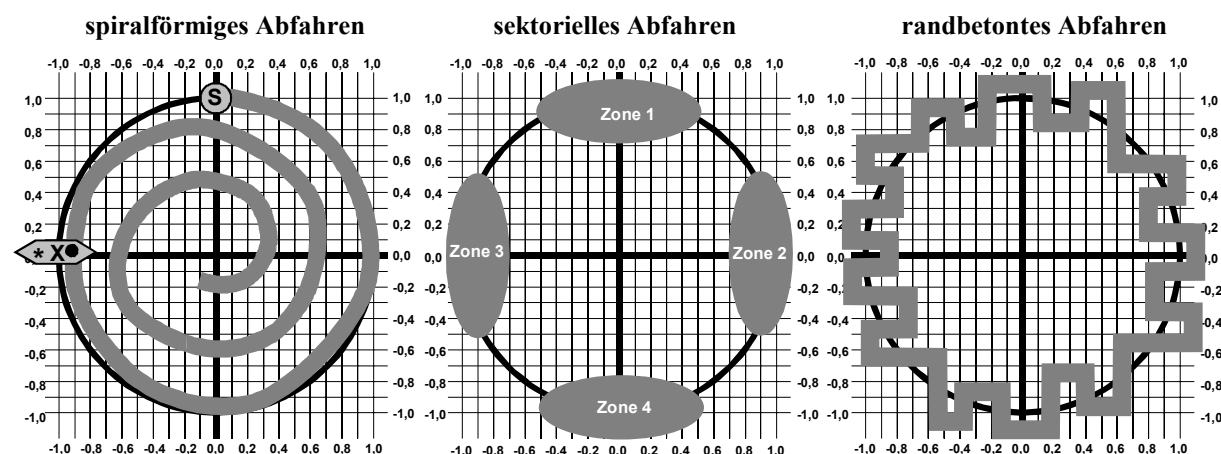
bevorzugt einige ausgeprägte Matrixkristalle am Rand der Probenspots. Vergleicht man die Präparationen mit DHB in den verschiedenen Lösungsmitteln, so ergeben sich aus den ausgeprägten Matrixkristallen wiederum die besten Spektren, vor allem hinsichtlich der Intensität. Bei 90% Acetonitril traten, in abgeschwächter Form, Probleme analog der Matrix-Präparation mit Acetonlösungen auf. Für alle später durchzuführenden Versuche erscheint ein Anteil von 50% Acetonitril für DHB-Lösungen im Hinblick auf Spektrenqualität (Intensität, Auflösung), morphologische Eigenschaften für die automatische MALDI-MS Messung (etwas feinere Matrixkristalle der Präparation) und Handhabung beim Pipettieren (Verflüchtigung, Oberflächenspannung) am geeignetsten.

Ein zusätzliche Verbesserung wurde häufig bei zweimaliger Auftragung der DHB-Matrix beobachtet. Hier zeigt die Präparation eine noch ausgeprägtere Homogenität, was bei der automatischen MALDI-MS im Durchschnitt oft kürzere Messzeiten pro Probe erbrachte. Zudem konnte eine höhere Erfolgsquote bei der Messung festgestellt werden, d.h. es wurden weniger Messungen abgebrochen, obwohl sich tatsächlich Analyten in der Probe befanden. Nachteil einer zusätzlichen Matrixauftragung ist natürlich die verlängerte Präparationszeit.

3.2.1.2 Optimierung des Abtastungsrasters der Spots

Bei der automatischen Messung läuft der Laser jeden Probenspot nach einem vorher festgelegten Raster ab. Der Spot wird dabei als Einheitskreis mit dem Radius 1 angesehen und die anzufahrenden Punkte werden als x-y-Paare in einer Textdatei zur Verfügung gestellt. Für CHCA-Präparationen wird in den meisten Fällen ein mehr oder weniger stark ausgeprägtes spiralförmiges Raster verwendet. In Abbildung 78 (links) ist ein solches Raster schematisch dargestellt.

Abbildung 78: Abtastungsraster des MALDI-Targets für die AutoXecute Messung

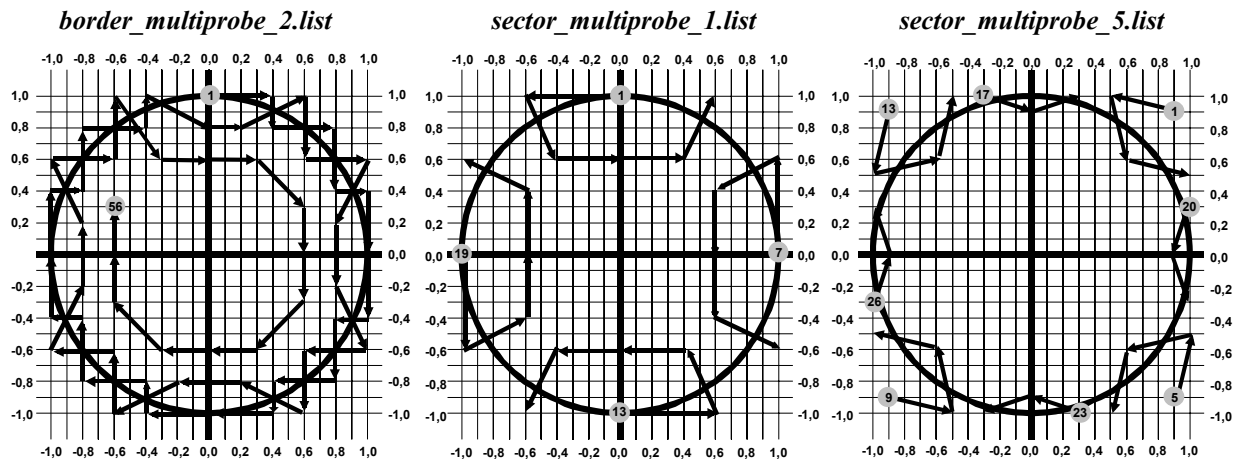


Die automatische Messung ist bei homogener Verteilung der Probe, wie im Falle der CHCA-Präparation, mit diesem Raster zumeist erfolgreich. Für die unregelmäßigen DHB-Präparationen erwies sich dies Raster als ungünstig. Können nach einer bestimmten Anzahl von Schüssen auf einem Spot keine Spektren akkumuliert werden, die den in der Methode aufgestellten Kriterien genügen, so wird die Messung für diesen Spot abgebrochen. Mit dem spiralförmigen Raster erwies sich diese Abbruchrate in den meisten Tests als zu hoch. Ein in den Experimenten oft beobachtetes Verhalten beim Abfahren der Spots kann dieses Verhalten bei der DHB-Matrix erklären. Erfolgt beispielsweise der Start der Spotmessung bei (S) in Abbildung 78 und liegt der nächste erreichbare Kristall der Matrix am Punkt (X), so wird über einen großen Zeitraum an vielen Punkten (je nach Feinheit des Rasters) kein Spektrum aufgenommen. Möglicherweise erfolgt bereits hier der Abbruch der Messung. Aber selbst wenn der Kristall bei (X) angefahren wird, ist nicht sichergestellt, dass hier auch tatsächlich ein gutes Spektrum erhalten wird. Der Erfolg hängt dann immer noch davon ab, ob der Kristall extreme *hot spots* aufweist und wie diese über dem Matrixkristall verteilt sind. In Abbildung 78 führt ein *hot spot* an der mit ● markierten Stelle bei Abfahren zu einer erfolgreichen Messung. Liegt der In der *hot spot* jedoch bei *, so bleibt die Messung erfolglos. Praxis haben sich für die automatischen Messungen der DHB-Präparationen ein sektorielles beziehungsweise ein randbetontes Abfahren der Probenspot als besser erwiesen. Das in Abbildung 78 dargestellte Sektoren-Raster teilt den Spot in 4 Zonen auf. Jede dieser Zonen wird gleichmäßig abgefahren. Entworfen Raster, die nach diesen Zonenmodell bei DHB-Präparationen gute Resultate lieferten sind:

- ❖ *sector_multiprobe_1.list* mit maximal 24 Rasterpunkten und
- ❖ *sector_multiprobe_5.list* mit maximal 28 Rasterpunkten

Die Zonenmodelle berücksichtigen dabei auch, dass die DHB-Matrix vorwiegend in den Randbereichen kristallisiert. Noch stärker auf diese Eigenschaft der DHB-Präparationen geht das randbetonte Abfahren des Spots mittels „*border_multiprobe_2.list*“ ein. Dieses Raster besitzt mit 56 eine sehr große Anzahl von Rasterpunkten, führte jedoch in nahezu allen Fällen bei der Messung zum Erfolg. Ein wesentlicher Nachteil gegenüber den beiden oben genannten Modellen zum sektoriellen Abfahren der Spots liegt in der zum Teil sehr langen Messzeit bei Verwendung dieses Rasters. Die folgenden Abbildungen zeigen den Verlauf der Spotabtastung für die drei erwähnten Raster.

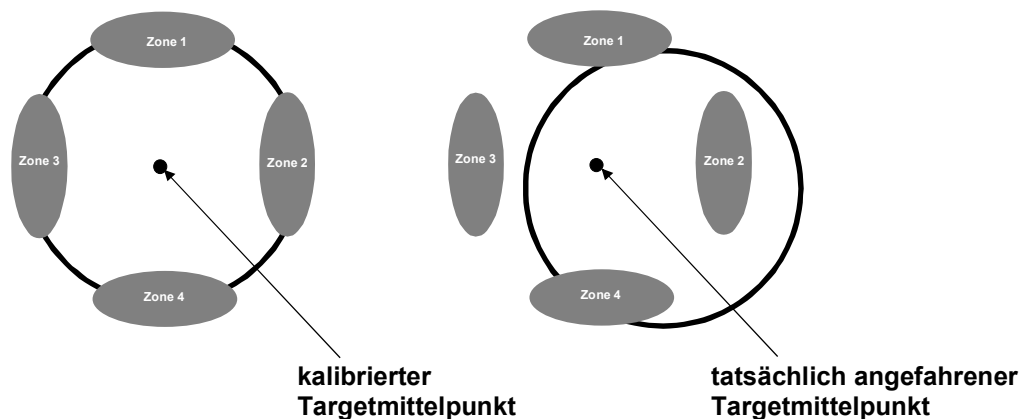
Abbildung 79: Abtastungsraster für DHB-Präparationen



3.2.1.3 Kalibrierung des MALDI-Targets für die AutoXecute

Für die automatische Messung muss eine exakte Kalibrierung jeder einzelnen Spotposition des MALDI-Targets durchgeführt werden, denn nur so kann sichergestellt werden, dass bei der Messung die im Raster eingetragene Position auch tatsächlich präzise angefahren wird. Trotz genauer Kalibrierung zeigte sich im Lauf der automatischen Messserien eine Verschiebung beim Anfahren der kalibrierten Spotmittelpunkte. Damit verschiebt sich aber auch das abgefahrte Raster auf dem Spot, wie dies in Abbildung 80 für das sektorielle Raster gezeigt ist.

Abbildung 80: Auswirkung des Hysterese-Effekt bei automatischer Messung mit dem sektoriellen Raster



Im Falle einer homogenen Matrixpräparation ist diese Verschiebung im Bezug auf die Messung relativ unkritisch. Für die sehr lokal am Spotrand kristallisierende DHB-Matrix ist die verschobene Lage des Rasters jedoch nur noch bedingt zur Messung geeignet. Bei einer

Kristallisation, wie sie sich im Falle der DHB-Matrix ergibt, zeichnet sich die beobachtete Verschiebung in einigen Fällen für den Misserfolg der Probenmessung verantwortlich. Laut Hersteller (telefonische Mitteilung) stellt diese Verschiebung einen bekannten Hysterese-Effekt dar, der durch Erwärmung eintritt. Für die x-y-Verschiebung des Targets bei der SCOUT 26 Ionenquelle lässt sich diese Hysterese nicht vermeiden.

3.2.1.4 Einzelaspekte der optimierten AutoXecute Methode

Um die automatische Messung zu starten, wird die Datei „*SAMPLES.TXT*“ nach Aufruf der AutoXecute in die Probenliste importiert. Anschließend kann mit der Messung sofort begonnen werden. Für jede Probe der Probenliste werden die in „*SAMPLES.TXT*“ festgelegten Parameter zur Messung verwendet. AutoXecute verwendet dabei die Einträge in den Spalten 1 und 27 bis 31 (Tabelle 64). Durch die ausschließliche Verwendung von MALDI-Targets der Geometrie SCOUT26 ist der Eintrag in Spalte 31 für alle Proben bindend. Die Einträge für „*Spectrum_Directory*“ (Spalte 30) und „*Spectrum_Filename*“ (Spalte 29) legen lediglich den Speicherort (Verzeichnis und Dateiname) der aufgenommenen Spektren fest. Der entscheidende Parameter für die automatische Messung der Proben steht in der Spalte „*AutoX_Method*“ (Spalte 28). Auf einzelne Aspekte der zur Messung der Leitersequenzen erarbeiteten Methode „*multiprobe_5*“ soll im folgenden kurz eingegangen werden. Die Einstellungswerte für „*multiprobe_5*“ stellen ein aus allen aufgenommenen Spektren empirisch erarbeitetes Optimum im Hinblick auf die MALDI-MS Messung von DHB präparierten Proben dar. Die im gezeigten Einstellungen betreffen nur diejenigen Parameter im Methoden Editor der AutoXecute, die für die Messung der Proben und die Prozessierung der erhaltenen Spektren im Rahmen dieser Arbeit von Bedeutung sind. (Ergänzende Angaben finden sich in den Dateien der beiliegenden CD-ROM). Die Parameter der Methode wurden im Hinblick auf die Kriterien „Schnelligkeit der Messung“ und „Erfolg der Messung“ optimiert. Zu diesem Zweck wurde für alle Messungen mit dem *Fuzzy Control* Modul der AutoXecute Software gearbeitet. Für die *Fuzzy Control* selbst wurde die Standard-Kontrolldatei „*tof_default*“ verwendet.

Allgemeine Einstellungen (Reiter „General“):*General Settings:*Parameter File: ***multiprobe_1000-4500_20m***Fuzzy Cntrl File: ***tof_default***XTOF Macro: ***Multiprobe_Plot_TI***Laser Einstellungen (Reiter „Laser“):*Laser Power:*Fuzzy Control: ***ON***Weight: ***1.0***Initial Attenuation: ***55***Minimal Attenuation : ***40****Matrix Blaster :*Fire initially: ***10*** shots with attenuation: ***50***Einstellungen für die Spektrensummierung (Reiter „Summing“)*Spectra Summing:*Fuzzy Control: ***ON***Weight: ***1.0***Sum up: ***50 shots*** in: ***5*** shot steps*Spectra Judging:*Smoothig: ***OFF***Peak: ***resolution*** must be higher than: ***4000***Signal/Noise threshold: ***10***Noiserange: ***1000***Einstellungen für die rasterförmige Abtastung des Targets (Reiter „Moving“)*Position Movement:*Maximal allowed shot number at one position: ***20***Maximum Number of consecutively rejected trials: ***50***

Der für die Methode dabei verwendete XACQ-Parametersatz *multiprobe_1000-4500_20m.par* (auf CD-ROM, siehe Anhang E) entspricht dem allgemeinen Parametersatz 2 zur UV-MALDI-MS Messung von Peptiden (siehe 2.6.1 auf Seite 90). Allerdings ist der Messbereich auf $m/z = 1000-4500$ begrenzt. Die obere m/z -Grenze ist willkürlich als Ende des interessierenden Bereichs gewählt. Die richtige Wahl der unteren m/z -Grenze stellt jedoch einen kritischen Parameter dar. Im Bereich unter 1000 Da treten sicherlich Signale von Leiterpeptiden auf. Ebenso sind hier aber auch die Signale von Störsubstanzen, wie z.B.

Puffersalzen, zu finden. Die Automatik kann in der vorliegenden Form nicht zwischen Signalen peptidischen Ursprungs und solchen Störsignalen unterscheiden. Ist daher die untere Grenze des Messbereichs zu niedrig gewählt, werden auch Spektren summiert, die nur oder überwiegend solche Störsignale enthalten. Bei den durchgeführten Versuchen hat eine schrittweise Erhöhung der unteren Grenze um jeweils 100 Da, beginnend bei 700 Da gezeigt, dass erst ab 1000 Da solche Fehler bei der Spektrenaddition weitgehend vermieden werden. In der Praxis bedeutet dies allerdings auch, dass einige Leitersequenzen, die bis 1000 Da herunterreichen, nach dem Automatiklauf eventuell noch einmal manuell nachgemessen werden müssen, um weitere Sequenzinformationen zu erhalten. Das XTOF Makro „*Multiprobe_Plot_TI*“ (Datei „*Multiprobe_Plot_TI.aura*“) sorgt dafür, dass das Leiterspektrum durch die XTOF-Software ausgewertet, beschriftet und ausgedruckt wird.

Der gewählte Abschwächungsbereich für den Laser bei Messungen der Leitersequenzen ist mit 40-55 im Vergleich zur Messung reiner Peptide deutlich niedriger gewählt. Die Schwelle (Threshold) für die Ionenerzeugung bei Verwendung von DHB-Präparationen lag für reine Peptide auf dem verwendeten MALDI-MS System bei gleichen Einstellungen im Laser-Setup der XACQ-Software in der Regel zwischen 60 und 65. Für die Messung der Leiterpeptide, insbesondere in Kombination mit einer automatischen Messung, sollte die Anfangsabschwächung auf jeden Fall niedriger gewählt werden als die üblichen Schwellenwerte zur Ionenerzeugung. Eine Anfangsabschwächung für die AutoXecute Messung, die um 10-20% niedriger liegt als die „normale“ Abschwächung zur Ionenerzeugung mit DHB-Matrix, hat sich als geeignet erwiesen. Die Abschwächung kann bis zu 40% unter dem Schwellenwert liegen, darunter nimmt die Auflösung der Signale meist stark ab. Die Notwendigkeit einer höheren Laserintensität zur Erzeugung ausreichend intensiver Signale bei den Leitersequenzierungen lässt sich auf die Anwesenheit der Puffersalze zurückführen, die zu einer mehr oder minder starken Suppression der Peptidionen führen. Die Verwendung des „*Matrix Blaster*“ hat sich im Zusammenhang mit der Messung von Präparationen für die Leitersequenzierung, die ja in der Regel relativ viel Salz enthalten, daher ebenfalls als wichtig erwiesen. Gerade die bei den ersten Laserschüssen erhaltenen Spektren zeigen oft sehr intensive Störsignale und sind nicht repräsentativ für die Zusammensetzung der Probe. Durch den „*Matrix Blaster*“ werden im Ablauf der Messung an jeder Stelle des Rasters die oberen Schichten der Präparation zunächst durch 10 Schüsse bei niedriger Laserabschwächung (Einstellungswert 50) abgetragen. Erst die Spektren der folgenden Laserbeschüsse werden dann zur Beurteilung und Summierung durch die *Fuzzy Logic* herangezogen.

Die Optimierung der Parameter für die Spektrensummierung (*Spectra Summing*, *Spectra Judging*) und die rasterförmige Laserbewegung über das Target (*Position Movement*) kann nicht einzeln aufgeschlüsselt werden, sondern muss als Komplex im Ganzen erfolgen. Durch die eingestellten Parameter wurden bei den Versuchen folgende Kriterien erfüllt:

- ❖ Die geforderte Zahl der summierten Einzelspektren (50) garantiert in den meisten Fällen ein qualitativ ausreichendes Spektrum im Hinblick auf die Intensität der Peptidsignale.
- ❖ Schnelle Abtastung des Spotrasters durch den Laser auf der Suche nach einer geeigneten Stelle der Präparation; Bei Misserfolg auf einer Position des Rasters wird relativ zügig zur nächsten Position gewechselt.
- ❖ Ausreichende Qualität der summierten Spektren (Auflösung und Signal/Rausch-Verhältnis).
- ❖ Wird eine gute Stelle (oder ein *hot spot*) auf der Präparation gefunden, so erlaubt die Verwendung der *Fuzzy Logic*, dass hier genügend Spektren gesammelt werden können um auf die geforderte Summe (50 Einzelspektren) zu kommen.

Die Parameter für die Spektrensummierung und die Laserbewegung weisen auch optimale Werte für den gewählten Abschwächungsbereich des Lasers auf. Die Fuzzy Control nimmt die Abschwächung des Laser bei Misserfolg (= Spektrum wird nicht summiert) um jeweils 3 Einheiten zurück. Für jede Position des Rasters wird somit immer nahezu das gesamte Abschwächungsintervall durchlaufen bevor die nächste Position im Raster angefahren wird.

3.2.1.5 Kalibrierung

Die Messung der Kalibranden erfolgt nach dem gleichen Prinzip wie die Messung der Proben. Die zur Messung verwendete Methode „*multiprobe_cal_5*“ für die AutoXecute weist leichte Veränderungen im Vergleich zu „*multiprobe_5*“ auf. Diese Veränderungen betreffen zum einen die etwas höheren Anforderungen an das minimale Signal/Rausch-Verhältnis im Spektrum (Signal/Noise threshold: 50), sowie die insgesamt höhere Anzahl zu summierender Spektren (100). Die Veränderung dieser beiden Parameter reicht aus, um in allen getesteten Fällen Spektren zu erhalten, die eine anschließende Auto-Kalibrierung über erlauben. Diese Kalibrierung erfolgt über das Makro „*Multiprobe_Cal_5Punkt*“ (Datei „*Multiprobe_Cal_5Punkt.aura*“) in der XTOF-Software. Der Aufruf für dieses Makro ist in der Methode „*multiprobe_cal_5*“ ebenfalls gegenüber „*multiprobe_5*“ abzuändern.

3.2.1.6 *Der Einfluss von Salzen*

Die bevorzugte Wahl der DHB-Matrix für die Probenpräparation bei der enzymatischen Sequenzierung basierte zum einen auf weit weniger stark ausgeprägten Suppressionseffekten der Leiterpeptide untereinander, zum anderen auf der höheren Toleranz gegenüber der Anwesenheit verschiedener Salze. Bei den durchgeführten Leitersequenzierungen treten vor allem folgende Salze in größeren Mengen in den MALDI-Präparationen auf:

- ❖ Natriumcitrat: findet in verschiedenen Enzymlösungen als Puffersubstanz in einer Konzentration von 0,1 – 13 mM Anwendung
- ❖ Tris: findet in verschiedenen Enzymlösungen als Puffersubstanz in einer Konzentration von 5 – 100 mM Anwendung
- ❖ Ammoniumsulfat: stammt als Stabilisator aus den Stammlösungen der Enzyme APM und LAP und findet sich in den zur Sequenzierung angewendeten Verdünnungen in einer Konzentration von 5 – 250 mM wieder.

Die DHB-Matrix zeigt bei diesen Salzkonzentrationen nicht nur eine geringere Peptidsignalunterdrückung, sondern auch Vorteile bezüglich der resultierenden Matrix-Morphologie der salzhaltigen Präparation. Präparationen mit CHCA-Matrix besitzen nämlich nur in Abwesenheit von Salzen generell die in Abbildung 77 gezeigte Homogenität. Abbildung 81 und Abbildung 82 zeigen Präparationen von CHCA und DHB nach Behandlung mit unterschiedlich konzentrierten Tris- und Natriumcitrat-Lösungen.

Während die CHCA-Präparation bei 10mM Tris noch keine Veränderungen zur Präparation ohne Salz zeigt, werden die Matrixkristalle bereits durch 50mM Tris stark angegriffen (hell leuchtende Stellen bei (B)). Obwohl die Präparation hier noch ein sehr homogenes Aussehen zeigt, sind die Peptidsignale bei gleicher Peptidmenge in den erhaltenen Spektren deutlich intensitätsschwächer. Bei 100mM Tris treten nur noch vereinzelt gut ausgeprägte CHCA-Matrixkristalle auf, z.B. bei (C). Eine automatische Messung ist hier jedoch kaum mehr möglich. Die DHB-Matrix verändert ihr Erscheinungsbild dagegen bei den entsprechenden Salzkonzentrationen bis 100mM nur unwesentlich. Spektren, die von den ausgeprägt kristallinen Stellen der Matrix der DHB-Präparation aufgenommen wurden, zeigen auch bei größeren Salzmengen in der Präparation eine gute Peptidsignal-Intensität. Bei 100mM Tris lassen sich zwei Bereiche unterschiedlicher Morphologie unterscheiden. Spektren, die in beiden Bereichen aufgenommen wurden, lassen den Schluss zu, dass sich der überwiegende Teil des Tris im feinkristallinen Bereich (E) befindet. Im Bereich (F) werden weiterhin gute bis sehr gute

Peptidspektren erhalten. Eine analoge Situation stellt sich bei Präparationen mit Citrat dar (Abbildung 82). Präparationen von CHCA und DHB in Anwesenheit üblicher Mengen Ammoniumsulfat (5 –250 nmol) liefern bei der N-terminalen Sequenzierung für die Messung in DHB ebenfalls bessere Ergebnisse. Im Bezug auf die automatische Messung kann bei Anwesenheit von Ammoniumsulfat, z.B. aus den verwendeten Aminopeptidasen APM und LAP, bis zu 100nmol Ammoniumsulfat in der Präparation ein Vorteil für DHB-Matrix gegenüber der CHCA-Matrix festgestellt werden. Bei größeren Mengen an Ammoniumsulfat wird auch die Morphologie der DHB-Präparation für eine automatische Messung zunehmend ungünstiger, wie Abbildung 83 zeigt.

Abbildung 81: Präparationen mit CHCA-Matrix (links) und DHB-Matrix (rechts) nach Applikation von 1µl der angegebenen Tris-HCl-Lösung (pH 8,2)

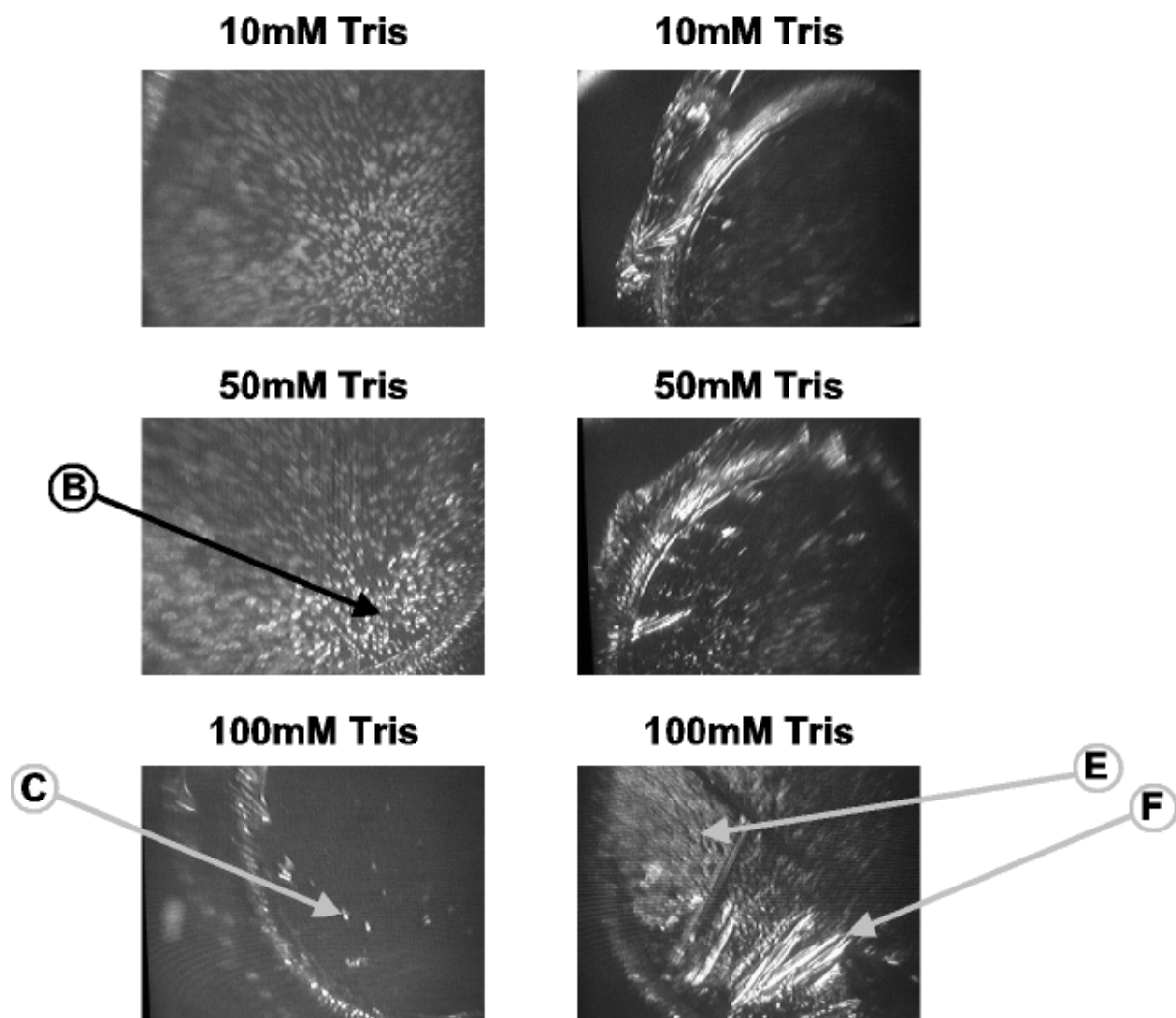


Abbildung 82: Präparationen mit CHCA-Matrix (links) und DHB-Matrix (rechts) nach Applikation von 1µl der angegebenen Natriumcitrat-Lösung (pH 6,0)

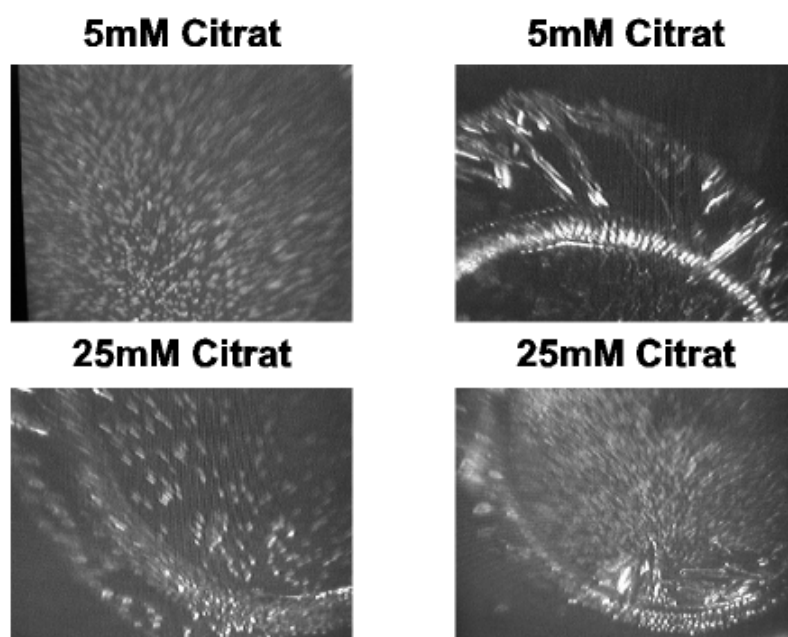
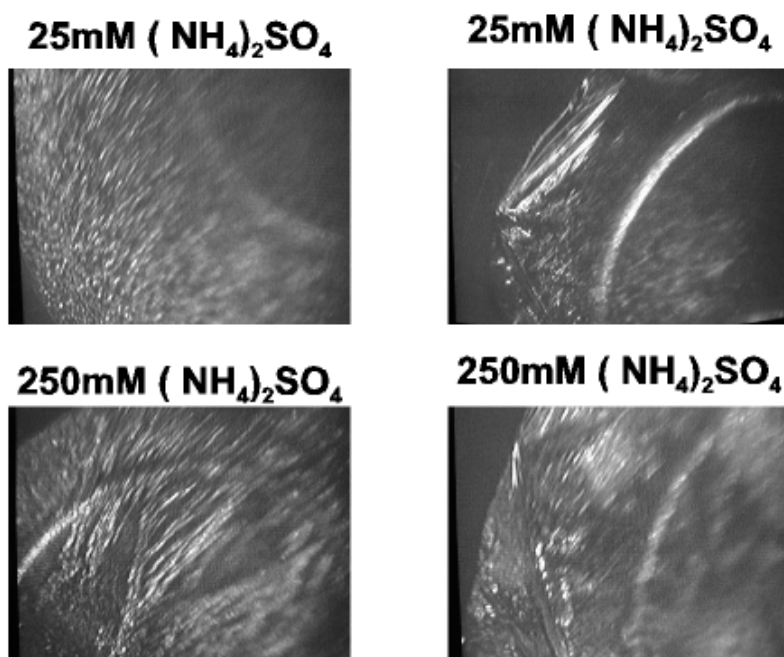


Abbildung 83: Präparationen mit CHCA-Matrix (links) und DHB-Matrix (rechts) nach Applikation von 1µl der angegebenen Ammoniumsulfat-Lösung



3.3 Ergebnisse für unterschiedliche Stufen der Automatisierung

Eine Bewertung der Automatisierung mittels MultiPROBE II und AutoXecute kann auf unterschiedlichen Automatisierungsstufen erfolgen. Nachfolgend werden Resultate bei zunehmenden Grad der Automatisierung in den analytischen Teilbereichen 2 und 3 (Abbildung 72) mit der manuellen Probenbearbeitung verglichen. Folgende Stufen der Automatisierung wurden untersucht:

Tabelle 65: Untersuchte Automatisierungsstufen

untersuchte Automatisierungsstufe	Beschreibung
<i>Automatisierungsstufe 1</i>	<ul style="list-style-type: none"> ❖ automatische Peptidprobenpräparation für MALDI-MS ❖ keine enzymatische Sequenzierung ❖ manuelle MALDI-MS Messung
<i>Automatisierungsstufe 2</i>	<ul style="list-style-type: none"> ❖ automatische Peptidprobenpräparation für MALDI-MS ❖ keine enzymatische Sequenzierung ❖ automatische MALDI-MS Messung
<i>Automatisierungsstufe 3</i>	<ul style="list-style-type: none"> ❖ automatische Peptidprobenpräparation für MALDI-MS ❖ mit enzymatischer Sequenzierung ❖ manuelle MALDI-MS Messung
<i>Automatisierungsstufe 4</i>	<ul style="list-style-type: none"> ❖ automatische Peptidprobenpräparation für MALDI-MS ❖ mit enzymatischer Sequenzierung ❖ automatische MALDI-MS Messung

3.3.1 Automatisierungsstufe 1

Für die Tests in den Automatisierungsstufen 1 und 2 wurden zunächst Lösungen von 22 synthetischen Peptiden herangezogen, die mit Massen von ca. 600-4100 Da den typischen Bereich proteolytischer Fragmente abdecken (Tabelle 66). Zusätzlich wurde eine Mischung aus fünf Kalibranden (Angiotensin 2, Bradykinin, Angiotensin 1, Neurotensin und ACTH Fragment 18-39) aufgetragen. Die automatischen MALDI-Präparationen zeigten als Resultat bei manueller MALDI-MS Messung keinerlei Nachteile gegenüber entsprechenden manuell erstellten Präparationen. Der Geltungsbereich dieser Aussage ist jedoch bezüglich einiger, für die MALDI-Präparationen kritischer Parameter, wie Lösungsmittelzusammensetzungen oder Pipettivolumina eingeschränkt:

- ❖ minimales Pipettiervolumen: 0,5µl
- ❖ Acetonitrilgehalt der pipettierten Lösungen: 0% – 70%

Das Pipettieren kleinerer Volumina oder von Lösungen mit höherem Acetonitrilgehalt ist jedoch für die Methode *MALDI-LS 1.42s* auch im Hinblick auf die Erfordernisse bei Proteinsequenzierungen nicht erforderlich.

Tabelle 66: synthetische Peptide für die automatische Präparation; Automatisierungsstufen 1 & 2

Peptid	Sequenz	Masse [M+H] ⁺ _{mono}
P5-603	RESLV	603,3466
P6-748	LSRFIL	748,4722
Dynorphin A Fragment 1-6 (Dyn-6)	LKWDNQ	803,4052
P6-841	YRVYIK	841,4946
Angiotensin 2	DRVYIHPF	1046,5423
Bradykinin	RPPGFSPFR	1060,5693
Nα-Ac-P12-1204	Ac-GASSGPTIEEVD	1203,5381
P10-1259	NGFRRTNEHK	1258,6405
Angiotensin 1	DRVYIHPFHL	1296,6853
Dynorphin A Fragment 1-10 (Dyn-10)	IRPKLKWDNQ	1297,7382
Substance P	RPKPQQFFGLM-NH ₂	1347,7361
Neurotensin	(pyro) ELYENKPRRPYIL	1672,9176
Dynorphin A Fragment 1-13 (Dyn-13)	LRRIRPKLKWDNQ	1723,0245
P18-2107	KLMDLDVEQLGIPEQEYS	2107,0270
Dynorphin A (Dyn-17)	YGGFLRRIRPKLKWDNQ	2147,1992
Adrenocorticotrophic Hormon (ACTH) Fragment 18-39	RPVKVYPNGAEDESAEAFPLEF	2465,1989
P24-2722	FEQRVNSDVLTVSTVNSQDQVTQK	2722,3646
P31-3467	YSDMREANYIGSDKYFHARGNYDAAKRGPGG	3466,5909
Insulin B-Kette, oxidiert	FVNQHLCGSHLVEALYLVCGERGFFYTPKA	3494,6513
P31-3649	YIFPVHWQFGQLDQHPIDGYLSHTELAFLRA	3648,8392
P37-4098	GYPPQQGYPPQQGYPPQQGYPPQQGYPPQQGYPPQQG	4097,8847

Für alle 22 Peptide und die Kalibrandenmischung wurden nach manueller und automatischer Präparation bei einer anschließenden manuellen (!) MALDI-MS Messung vergleichbare Resultate bezüglich Intensität und Auflösung der Peptidsignale erzielt. Präparationsfehler, wie z.B. ein fehlerhaftes Absetzen einer Lösung, konnten im Rahmen dieser Tests im Mittel maximal bei einer von 23 Präparationen (22 Probenspots + 1 Kalibrandenspot) beobachtet werden. Die Erfolgsquote der automatischen Präparationen lag somit bei ca. ≥ 96%. Querkontaminationen einzelner Proben konnten bei der MALDI-MS Messung nicht festgestellt werden.

Als Erweiterung zur Untersuchung synthetischer Peptide wurden auf dem MultiPROBE II System in analoger Weise alle Fraktionen aus fünf verschiedenen HPLC-Läufen von Protein-

spaltungen mit der Methode *MALDI-LS 1.42s* präpariert. Dabei handelte es sich um Trypsinspaltungen der Proteine Aldolase, Myoglobin und Schweineserum Albumin (50 bis 200pmol Protein-Ausgangsmaterial). Es ergab sich ein nahezu identisches Bild wie für die synthetischen Peptide: Von insgesamt 211 präparierten Fraktionen wurden 7 Fraktionen fehlerhaft präpariert (misslungenes Absetzen einer Lösung auf dem MALDI-Target), was einer Fehlerquote von nur ca. 3% entspricht. Sofern die Proben jedoch korrekt präpariert wurden, konnten in den Spektren der anschließenden MALDI-MS Messung (manuell) wiederum keine Unterschiede zu den entsprechend manuell präparierten Fraktionen festgestellt werden.

Die mittleren Bearbeitungszeiten für einzelne Schritte der Präparation mit der Methode *MALDI-LS 1.42s* sind in Tabelle 67 aufgelistet. Die WinPREP Software legt für jede abgearbeitete Probenliste einen Report in der Accessdatenbank „*MultiPROBE.mdb*“ an. Die genannten Bearbeitungszeiten wurden als Mittelwerte aus diesen angelegten Reporteinträgen berechnet.

Tabelle 67: Zeitaufwand für einzelne Arbeitsschritte in der Methode *MALDI-LS 1.42s*

Arbeitsschritt	Zeitaufwand
Verteilungsschritte für Enzyme	minimal: 80 – 90s (nur 1 Enzym) maximal: 100 – 110s (3 Enzyme)
jeder andere Verteilungsschritt (z.B.: Pufferlösungen)	70 – 80s / Verteilungsschritt
Probenauftragung ¹	14 – 16s / Probe ²
Matrixauftragung ¹	14 – 16s / Probe ²
Reagenzauftragung	14 – 16s / Probe ²
Auftragung von Puffer 1 oder Puffer 2	14 – 16s / Probe ²
Enzymauftragung	9 – 10s / Probe ³
Auftragung von Puffer 3 oder Puffer 4	9 – 10s / Probe ³
Auftragung von Kalibrand + Matrix für den Kalibrand	110 – 120s

¹ nicht für den Spot mit den Kalibranden

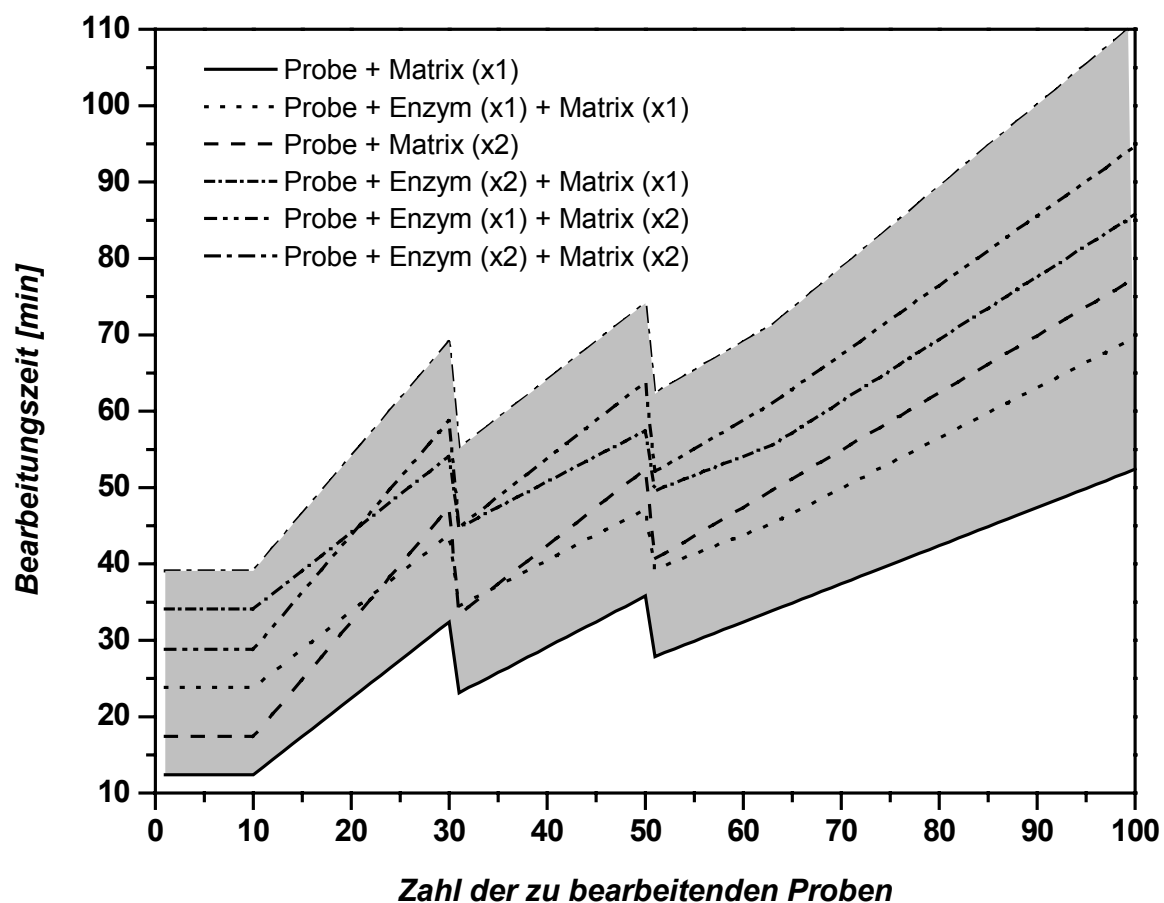
² Pipettierschritte mit zwischengeschaltetem organischen Waschschrift

³ Pipettierschritte ohne zwischengeschaltetem organischen Waschschrift

Aus diesen Daten kann für die jeweilige Anwendung der Zeitbedarf für die Probenpräparation abgeschätzt werden. Zu berücksichtigen sind dabei zusätzlich die über die Datei „FLAG PARAMETERS.TXT“ festgelegten Reaktionszeiten und die im Programm aus den Laufzeitvariablen berechneten Trocknungszeiten für Proben und Matrices auf dem Target. In

Abbildung 84 sind ungefähre Bearbeitungszeiten für Präparationen mit bis zu 100 Proben mit der Methode *MALDI-LS 1.42s* gezeigt. Der diskontinuierliche Verlauf bei 10, 30 beziehungsweise 50 Proben hängt mit der im Programm durchgeführten Berechnung der Trocknungszeiten zusammen. Ab 50 Proben läuft die Bearbeitungszeit durchgehend linear.

Abbildung 84: Abschätzung der Bearbeitungszeiten für verschiedene Präparation von bis zu 100 Proben mit der Methode *MALDI-LS 1.42s*



Anmerkung: (x1) steht für eine einmalige Auftragung, (x2) für eine doppelte Auftragung;

Die Bearbeitungszeiten in Abbildung 84 wurde für eine Verdauzeit mit Enzym #1 (Digest Time Enzyme #1) von 10min berechnet. Die gezeigten Kurven geben nur die Präparationszeiten für sechs einfache Fälle wieder, in denen noch dazu alle Proben gleich behandelt werden. Zu den real gemessenen Werten ergibt sich eine Abweichung von ca. 5-10%. Allerdings liefert das Modell eine gute Möglichkeit auch in anderen Fällen die Bearbeitungszeit komplexerer, aber vergleichbarer Präparationen auf maximal 10-20% Abweichung genau abzuschätzen.

3.3.2 Automatisierungsstufe 2

Als nächster Schritt wurde in der Automatisierungsstufe 2 getestet, inwiefern eine automatische MALDI-MS Messung mittels AutoXecute mit den aus der Automatisierungsstufe 1 stammenden Probenpräparationen (DHB-Matrix !) realisierbar ist. Die Messungen mit der Methode *multiprobe_5* lieferten für die 22 synthetischen Peptide die in Tabelle 68 gezeigten Resultate. Die untere m/z-Grenze im Spektrum wurde in verschiedenen Versuchen zwischen $m/z = 600$ und $m/z = 1000$ variiert. Die Spektren wurden aus DHB-Präparationen erhalten, in denen der Acetonitrilgehalt der Matrixlösung zwischen 30 und 70% lag. Es wurden sowohl Präparationen mit einfacher, als auch mit doppelter Matrixauftragung gemessen.

Tabelle 68: Resultate der automatischen MALDI-MS Messungen von synthetischen Peptiden

aufgetragene Peptidmenge [fmol]	maximale Fehlerquote [%]	Messzeit [min]	Messzeit pro Spot [s]
500	18%	ca. 60-75	ca. 160-205
250	14%	ca. 100-120	ca. 270-330
100	23%	ca. 85-165	ca. 230-450s

Alle Kalibrierungen erfolgten über eine quadratische Regression. Die mittlere Abweichung betrug dabei (15 ± 15) mDa bei automatischer Messung und ist damit vergleichbar zu (18 ± 18) mDa bei manueller Messung. Alle Messungen der Kalibranden verliefen erfolgreich, allerdings erfolgte in 2 von 7 Fällen die anschließende, automatische Berechnung der Kalibrierfunktion in der XTOF-Software aufgrund eines Fehlers bei der Peakselektion nicht richtig. Damit war in allen nachfolgenden Probenspektren die Angabe der Massen ebenfalls fehlerhaft. Eine manuelle Nachkalibrierung war also teilweise erforderlich.

Bei der manuellen Messung konnten für alle Peptide im Bereich von 500 bis 100fmol Peptidsignale in den jeweiligen Spektren beobachtet werden (entspricht 0% Fehlerquote). Bei manueller Abarbeitung der Proben betrug die mittlere Messzeit bei den Proben mit 250fmol Peptid zum Vergleich 90min (ca. 240-250s pro Spot) – allerdings im Gegensatz zur automatischen Messung ohne Spektrenbearbeitung und –ausdruck. Die angegebene Fehlerquote bezeichnet Messungen, bei denen entweder im aufgenommenen Spektrum nicht der Peak des erwarteten Peptids zu sehen ist oder Messungen, bei denen überhaupt kein Spektrum akkumuliert werden konnte (Messung des Spots wurde abgebrochen). Die Fehlerquote steigt zwar mit abnehmender Probenmenge nicht sehr stark, allerdings nimmt die Messzeit pro Spot (Probe) deutlich zu. Zudem kann die Messzeit von einem Automatiklauf zum anderen bei

geringen Probenmengen sehr stark abweichen. Hierfür zeichnen sich insbesondere diejenigen Spots verantwortlich, bei denen die Akkumulation abgebrochen werden muss. Um allerdings eine geringe Fehlerquote bei der Messung zu garantieren, muss bei den Parametern in der AutoXecute-Methode ein Kompromiss zwischen kurzer Messzeit und geringer Fehlerquote eingegangen werden. In der Folge kann die Messzeit bei manchen „schlechten“ Spots bis zu 15min (= 900s) betragen. Bei Peptiden, die eine gute Signalintensität im Spektrum zeigen beträgt dagegen die Messzeit in der Regel nur maximal ca. 3min (= 180s).

Mit den gleichen Parametern wie für die synthetischen Peptide wurden auch automatische MALDI-MS Messungen der HPLC-Fractionen der oben genannten, tryptischen Proteinspaltungen durchgeführt. Gemessen wurde die bereits vorherigen Abschnitt angefertigten MultiPROBE II Präparationen dieser Proteinspaltungen.

Tabelle 69: Resultate der automatischen MALDI-MS Messungen von tryptischen Proteinspaltungen

Herkunft der HPLC-Fractionen: tryptische Spaltung von	aufgetragene Peptidmenge ¹ [fmol]	Gesamtzahl der gemessenen Fractionen (mit Kalibrierung)	Fehlerquote [%] ²	Messzeit pro Spot [s]
Myoglobin 200 pmol	1000-2000	36	3%	ca. 430s
Schweineserum Albumin 200 pmol	1000-2000	50	10%	ca. 300s
Schweineserum Albumin 50 pmol	250-500	45	4%	ca. 370s
Aldolase 50 pmol	250-500	43	7%	ca. 450s

¹ zur Berechnung siehe Erklärung im nachfolgenden Text

² Vergleich zur manuellen Messung der gleichen Präparation

Eine Quantifizierung der in den Fractionen enthaltenen Peptimengen über ASA und damit eine Bestimmung der Peptidkonzentrationen war nicht möglich – die Peptidmengen lag in allen Fällen unter der Nachweisgrenze. Somit kann lediglich eine Abschätzung der Peptidkonzentration in den Fractionen erfolgen. Setzt man für jeden Schritt der Aufarbeitung (Reduktion, Alkylierung, HPLC-Aufreinigung, gelelektrophoretische Trennung, Spaltung im Gel, Elution aus dem Gel, HPLC-Trennung der Peptide) der zum Teil unvollständig sein wird oder Probenverluste birgt, einen Verlust von ca. 5% pro Schritt an, so beträgt die Ausbeute in den Peptidfractionen noch maximal ca. 70% der Ausgangsmenge. Bezogen auf ein typisches

Fraktionsvolumen von 50µl pro HPLC-Fraktion und 0,5µl Auftragsvolumen bei der automatischen Probenpräparation ergeben sich auf dem Target folgende Absolutmengen:

- ❖ 200pmol Proteinausgangsmenge: 1 – 2 pmol Peptid
- ❖ 50pmol Proteinausgangsmenge: 250 – 500 fmol Peptid

Es ist jedoch anzunehmen, dass die tatsächlichen Mengen noch deutlich unter diesen Werten liegen. In der Summe lässt sich feststellen, dass auch für die automatische Messung der proteolytischen Spaltpeptide gute bis sehr gute Resultate mit geringer Fehlerquote erhalten werden. Maximal zwei Proben pro tryptischer Proteinspaltung erforderten eine manuelle Nachmessung, da die Messung des betreffenden Spots abgebrochen wurde.

3.3.3 Automatisierungsstufe 3

Die Automatisierungsstufe 3 beinhaltet den Schritt der enzymatischen Sequenzierung im Ablauf der Methode *MALDI-LS 1.42s*. Die so angefertigten Präparationen wurden mit den entsprechenden manuellen Sequenzierungen verglichen. Alle im Rahmen dieses Vergleichs notwendigen MALDI-MS Messungen wurden manuell durchgeführt. Damit entspricht die Automatisierungsstufe 3 einer vollständigen Automatisierung des Bereichs „Analytik Teil 2“ in Abbildung 72. Insgesamt wurden 88 Fraktionen aus Spaltungen der Proteine Aldolase und Myoglobin mit Trypsin, Chymotrypsin und Endoproteinase GluC (Phosphat) untersucht. Die Sequenzierungen beschränkten sich auf die Applikation eines Enzyms oder einer Enzymmischung pro Spot.

N-terminale Sequenzierungen erfolgten mit den Aminopeptidasen:

- ❖ AAP (5-10mU)
- ❖ API (4-7mU)
- ❖ APM (0,2-1mU)

C-terminale Sequenzierungen wurden mit den folgenden Carboxypeptidasen durchgeführt:

- ❖ CPY (3-6mU)
- ❖ CPP (0,25-0,5mU)
- ❖ CPY + CPP (3mU für CPY bzw. 0,25mU für CPP)

Tabelle 70 zeigt die statistische Auswertung der Sequenzierungsergebnisse.

Tabelle 70: Resultate der automatischen Sequenzierung proteolytischer Spaltpeptide

proteolytische Spaltung (Protein / Endopeptidase)	Zahl der sequenzierten Fraktionen	Abbaulänge ¹ / Sequenzlänge ² bei	
		automatischer Sequenzierung	manueller Sequenzierung
<i>N-terminale Sequenzierung</i>			
Aldolase / Trypsin	13	30 / 30	32 / 32
Aldolase / Chymotrypsin	21	65 / 51	60 / 53
Aldolase / GluC (Phosphat)	18	93 / 58	97 / 66
Myoglobin / Trypsin	13	19 / 19	30 / 30
<i>C-terminale Sequenzierung</i>			
Aldolase / Trypsin	17	32 / 30	39 / 33
Myoglobin / Trypsin	6	22 / 22	17 / 15
<i>Summe</i>			
Endopeptidasespaltungen aller untersuchten Proteine	88	261 / 210	275 / 229

¹ Summe der abgebauten Aminosäuren aller untersuchten Peptide bei Sequenzierungen dieser Spaltung

² Summe der abgebauten Aminosäuren, die Sequenzinformation bei dieser Spaltung liefern

Die gesamte Abbaulänge bei den automatischen Sequenzierung liegt nur um ca. 5% unter der Abbaulänge der vergleichbaren manuellen Sequenzierungen. Eine Betrachtung der Einzelergebnisse in Tabelle 70 legt jedoch die Schlussfolgerung nahe, dass dieser Unterschied nicht signifikant ist, da die erhaltenen Abbaulängen der automatischen Sequenzierungen nicht systematisch schlechter sind (siehe z.B. Spaltung der Aldolase mit Chymotrypsin und Spaltung des Myoglobins mit Trypsin). Die Qualität der MALDI-Leiterspektren bezüglich der monoisotopischen Signalauflösung und der Signalintensität, die zu den Werten dieser Untersuchung geführt hat, ist bei manueller und automatischer Präparation vergleichbar. Die erhaltene Sequenzlänge über alle sequenzierten Peptide der Automatisierungsstufe 3 liegt ebenfalls nur 8% unter der manuell ermittelten Sequenzlänge, wobei auch hier die automatische Sequenzierung nicht systematisch kürzere Sequenzlängen liefert.

Gemäß der Zahl untersuchter Fraktionen und getesteter Enzyme wurden insgesamt ca. 250 automatische Sequenzierungen durchgeführt. Die Fehlerquote bei der Präparation liegt auch unter Einbeziehung eines Sequenzierungsschritts nur bei 3% und ist damit nicht höher als in der Automatisierungsstufe 1.

3.3.4 Automatisierungsstufe 4

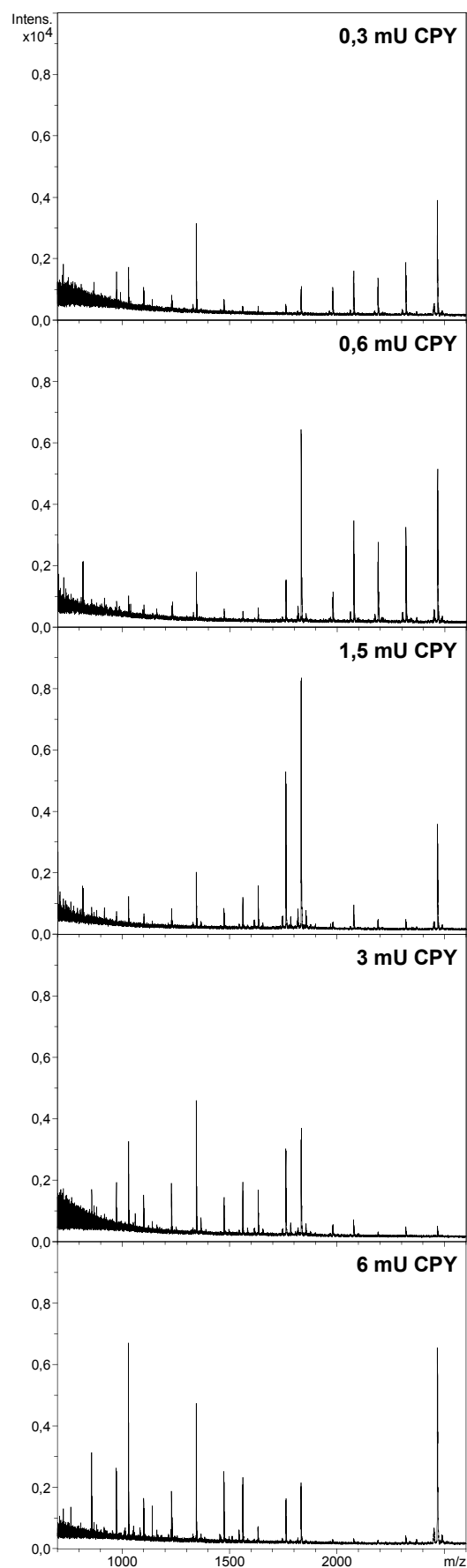
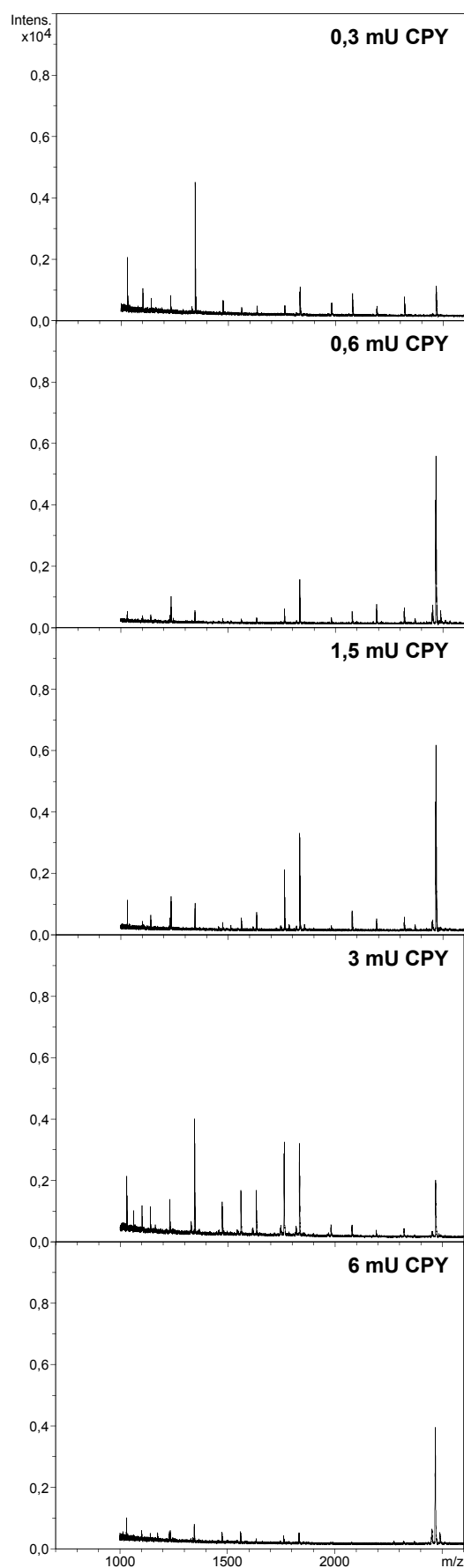
Die Automatisierungsstufe 4 entspricht einem vollautomatischen Ablauf der Leitersequenzierung von der Präparation der Proben bis einschließlich der MALDI-MS Messung. Es liegt somit eine vollständige Automatisierung der analytischen Teilbereiche 2 und 3 vor (siehe Abbildung 72).

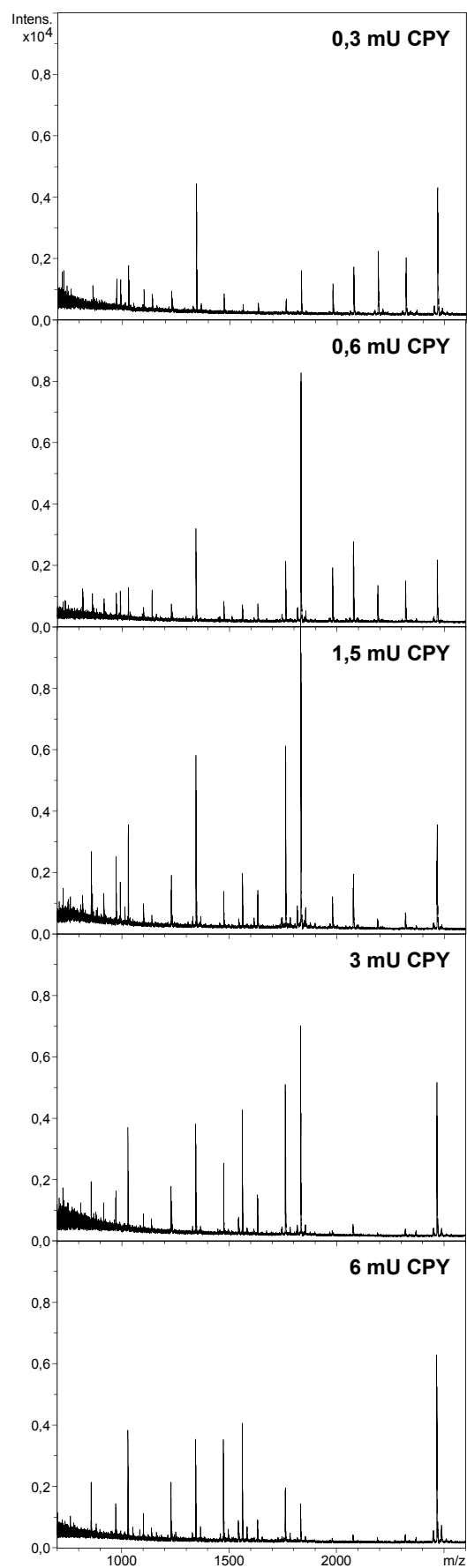
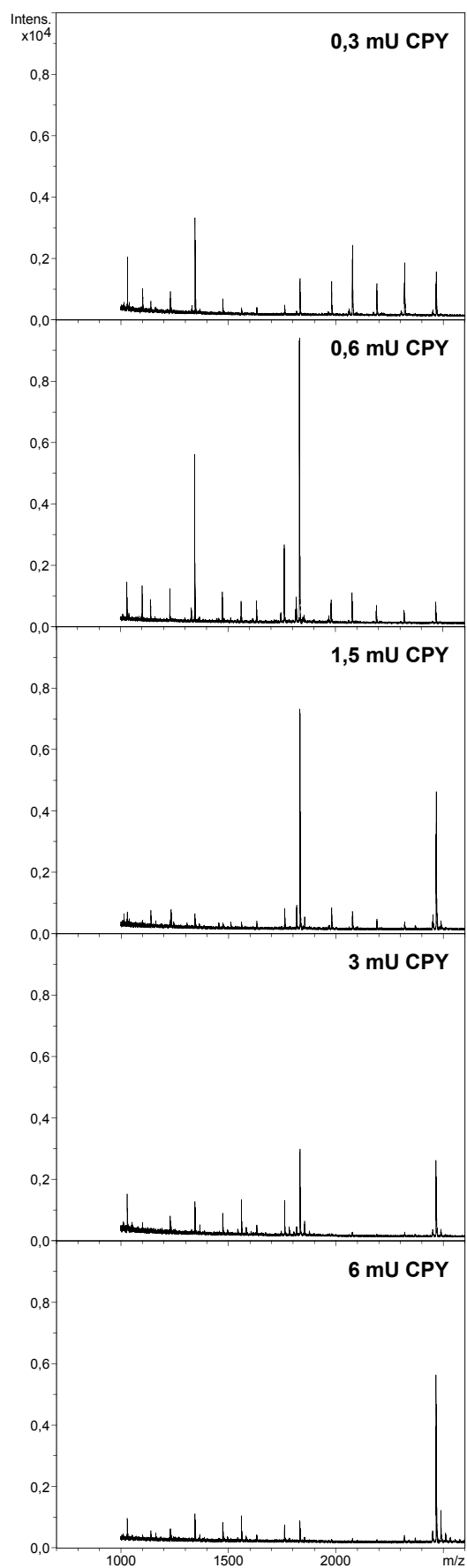
Für die Tests dieser Automatisierungsstufe wurden Sequenzierungsserien von ACTH Fragment 18-39 unter folgenden Rahmenbedingungen mit den Methoden *MALDI-LS 1.42s* (MultiPROBE II) und *multiprobe_5 / multiprobe_cal_5* (AutoXecute) durchgeführt:

- ❖ Sequenzierte Menge ACTH: 500fmol
- ❖ Sequenzierungstemperatur: 33°C
- ❖ CPY-Verdauzeit: 5min (Digest Time Enzyme #1 in *FLAG PARAMETERS.TXT*)
- ❖ DHB-Matrix Lösungsmittel: ACN/Wasser 50/50 (v/v) + 0,1% TFA
- ❖ Matrixauftragung: einfach (Multiple Matrix Addition = 0 in *FLAG PARAMETERS.TXT*)

Abbildung 85 zeigt die Ergebnisse der Automatisierungsstufe 4 und den Vergleich zur manuellen Bearbeitung für zwei Sequenzierungsserien. Die manuell bearbeiteten Proben wurden identisch zu den automatisch bearbeiteten Proben präpariert und gemessen. Aufgrund der erwähnten Limitierung des Massenbereichs in der AutoXecute Methode *multiprobe_5* endet in allen Spektren der automatischen Serien der Messbereich bei 1000Da. Die Einzelspektren der Serien zeigen für vollautomatische und manuelle MALDI-Leitersequenzierung eine gute bis sehr gute Übereinstimmung. Während die auftretenden Leiterpeptide bei Anwendung einer bestimmten Enzymaktivität für manuelle und automatische Bearbeitung gleich sind, zeigen sich wesentliche Unterschiede in den Signalintensitäten dieser Leiterpeptide. Eine nachträgliche, manuelle Messung der automatisch präparierten Proben zeigt, dass diese Intensitätsunterschiede auf die AutoXecute Messung zurückzuführen sind und nicht etwa auf eine qualitativ schlechtere Präparation. Bei manueller Messung der automatisch präparierten Proben lassen sich grundsätzlich vergleichbare Signalintensitäten zur manuellen Bearbeitung realisieren, da hier einzelne, *hot spots* der Präparation selektiver und präziser angefahren werden können.

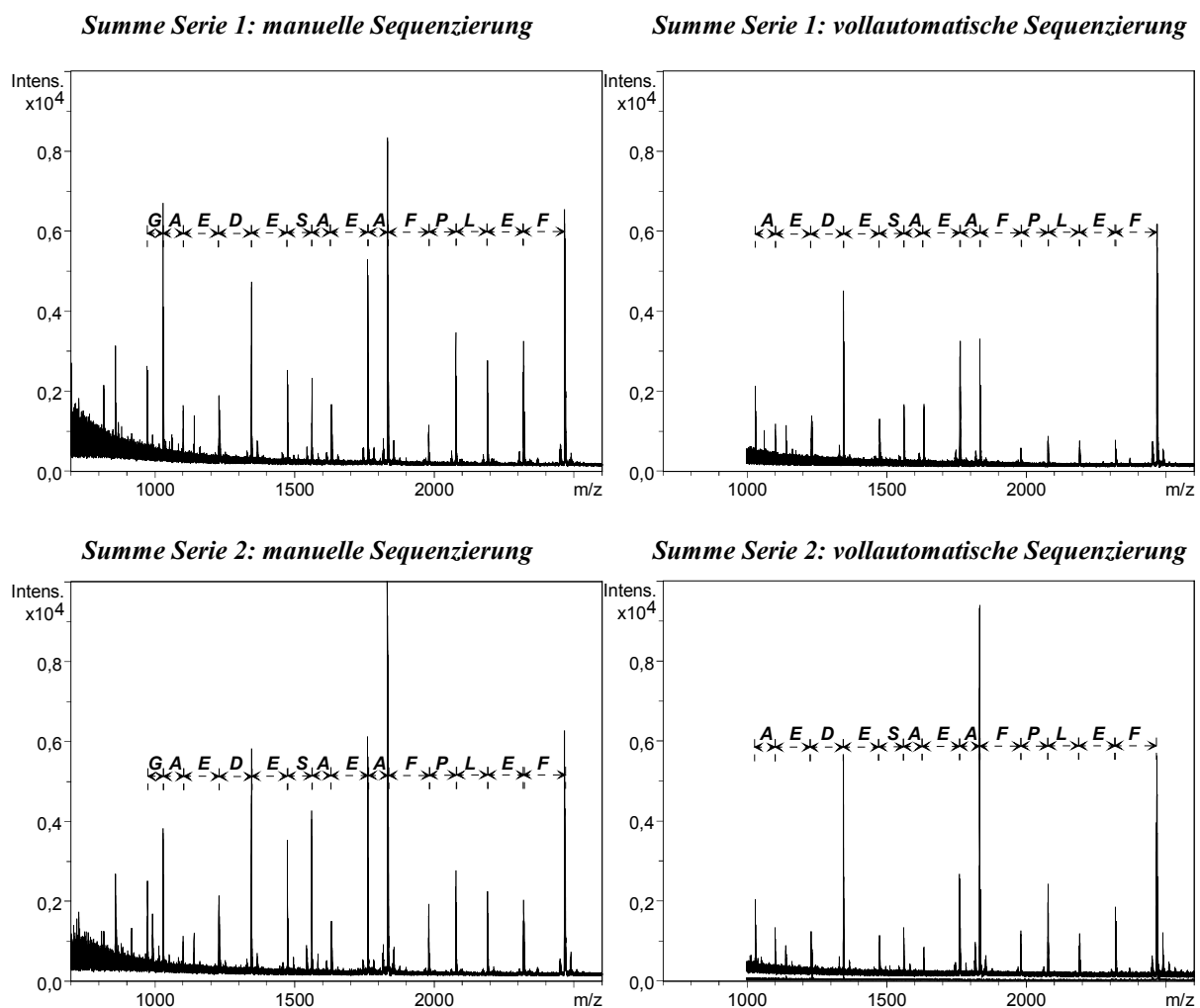
Abbildung 85 (folgende Seiten): Vergleich von vollautomatischer und manueller Sequenzierung von 500fmol ACTH 18-39 mit CPY.

Serie 1: manuelle Sequenzierung*Serie 1: vollautomatische Sequenzierung*

Serie 2: manuelle Sequenzierung*Serie 2: vollautomatische Sequenzierung*

Auch in der Summe (Abbildung 86) zeigt sich für die Mehrzahl der Leiterpeptide aus den Automatikläufen eine um etwa 50-75% geringere Signalintensität. Die Sequenz lässt sich dennoch auch bei den Spektren der vollautomatischen Sequenzierung deutlich und eindeutig auslesen.

Abbildung 86: Überlagerte Einzelspektren der ACTH-Sequenzierungsserien: Vergleich manueller und vollautomatischer Sequenzierung



Auf der anderen Seite ergaben sich weder Fehler bei der automatischen Präparation, noch bei der automatischen MALDI-MS Messung (Fehlerquote in beiden Fällen 0%).

Pro Serie wurden fünf Sequenzierungen, sowie zusätzlich eine Kalibrandenmischung und eine ACTH-Blindprobe gemessen. Im Mittel errechnet sich eine Messzeit von ca. 340s pro Probe. (Maximalwert ca. 410s pro Probe, Minimalwert ca. 270s pro Probe). Berücksichtigt man die Tatsache, dass sich die 500 fmol des aufgetragenen ACTH auf mehrere Leiterpeptide vertei-

len, so liegt die benötigte Messzeit pro Spot im Falle der Leitersequenzierung durchaus bei einem Wert, der mit den unter 3.3.2 (Kapitel II) angegebenen Werten für die automatische MALDI-MS Messung salzfreier Peptidproben vergleichbar ist.

4 Diskussion und Bewertung

Im Bezug auf die Übertragung der manuell ausgearbeiteten Sequenzierungsmethoden auf ein automatisiertes System zeigt sich, dass eine Bewertung für den Teil der Probenpräparation und der anschließenden Probenmessung getrennt erfolgen sollte.

Grundsätzlich ließen sich mit manuell aufgenommenen Massenspektren von den automatisch und manuell präparierten Proben qualitativ völlig vergleichbare Resultate erzielen, womit die Evaluierung des Teilbereichs der Probenpräparation hinsichtlich einer Automatisierung als erfolgreich zu bezeichnen ist. Auf der anderen Seite ergeben sich bei den automatischen MALDI-MS Messungen der DHB-präparierten Proben noch einige Probleme, die jedoch ihre Ursache zum großen Teil in der gegebenen Instrumentierung haben.

Die Diskussion für dieses Kapitel beinhaltet daher im wesentlichen, neben der Zusammenfassung einiger Charakteristika der entworfenen Automatisierung, technische Verbesserungsmöglichkeiten der ausgearbeiteten Methode. Da im Zuge der Ausarbeitung und Evaluierung der Methode nur Durchläufe mit maximal 50 Proben bearbeitet wurden, werden in diesem Abschnitt auch noch einige Punkte angesprochen, die im Zusammenhang mit der Bearbeitung größerer Probenzahlen von Bedeutung sind. Es sollen auch solche Verbesserungen diskutiert werden, die sich durch technische Weiterentwicklungen, welche jedoch im Rahmen dieser Arbeit noch nicht zur Verfügung standen, ergeben können.

4.1 Reaktionszeiten bei der Sequenzierung

Bezüglich einer frei wählbaren Spaltungszeiten ergibt sich ein unteres Limit, dass im allgemeinen durch die Zahl der in einem Schritt der Methode *MALDI-LS 1.42s* zu bearbeitenden Proben gegeben ist. Soll die Spaltung nach einem definierten Zeitintervall vor dem Antrocknen der Enzymlösung abgebrochen werden, so ist dies frühestens nach Beendigung des vorhergehenden Arbeitsschritts möglich. Bei 25-30°C und einer Auftragung von 1,0µl wässriger Enzymlösung beträgt die übliche Trocknungszeit von 4-5min. Eine Abschätzung über die experimentell ermittelten Daten in Tabelle 67 (S. 276) ergibt jedoch, dass bereits die Bearbeitung von 25 Proben für den Arbeitsschritt der Enzymauftragung ca. 4min in Anspruch nimmt. Die Optimierung der Kinetik kann daher im Fall einer zu schnellen Abspaltung der Aminosäuren nicht über die Verkürzung der Spaltungszeit, sondern nur über eine Reduzierung der Enzymmenge erfolgen. Eine Verkürzung der Reaktionszeit wäre für das MultiPROBE II System in der gegebenen Form nur bei Einzelabarbeitung der Proben

nacheinander möglich, d.h. für jede Probe werden alle Schritte zunächst vollständig durchgearbeitet, bevor mit der nächsten Probe fortgefahren wird. Ein solches Abarbeitungsschema ist jedoch gerade für größere Probenzahlen viel zu zeitaufwendig. Durch zwei voneinander unabhängig arbeitende Pipettierarme wäre eine Lösung gegeben, alle Reaktionszeiten in einem völlig flexiblen Rahmen ablaufen zu lassen.

4.2 Technische Verbesserungen im Teilbereich der automatischen Probenpräparation

Bei den gezeigten Präparationen mit maximal 50 Proben ergaben sich durch offen stehende Gefäße für Proben-, Matrix-, Puffer- und Kalibrandenlösungen noch keine Probleme. Die Präparationszeiten waren zu kurz, um in den Resultaten der Präparationen einen sichtbar, negativen Effekt durch Verdunstung von Lösungsmitteln zu verursachen. Bei einer größeren Anzahl von Proben und somit auch längeren Präparationszeiten könnten sich durch die offen stehenden Lösungen jedoch durchaus Probleme ergeben – insbesondere bei Lösungen mit hohem Anteil an flüchtigen, organischen Lösungsmitteln. Dies betrifft vor allem auch diejenigen Peptide in HPLC-Fractionen mit hohem Acetonitrilanteil und auch die Matrixlösungen. Bei HPLC-Fractionen, die hydrophobe Peptide enthalten, besteht bei der Verdunstung zudem die Gefahr von Adsorptionsverlusten an den Gefäßwänden. Bei der ausschließlichen Verwendung von flachen Mikrotiterplatten bietet sich die Verwendung einer dünnen Folie an, die erst unmittelbar vor der Flüssigkeitsentnahme von der Pipettiernadel durchstochen wird. Die Verwendung von 0,5ml Reaktionsgefäßen für Lösungen, von denen größere Volumina benötigt werden (z.B. Matrix- oder Enzymlösungen) ist dann jedoch mit dem MultiPROBE II mit der gegebenen Nadelform (*low volume tips*) nicht mehr möglich.

Die Präparation der Proben auf den offen stehenden MALDI-Targets, wie im vorliegenden Fall, ist im Bezug auf die Reproduzierbarkeit der Probenpräparation ebenfalls ein kritischer Punkt. Während in einem abgedeckten System bei manueller Präparation die Trocknungszeiten der Enzymlösungen, und somit die Sequenzierzeiten, gut reproduzierbar waren, beobachtet man beim offenen MultiPROBE System zum Teil trotz Thermostatisierung deutliche Schwankungen in den Trocknungszeiten von einer Experimentserie zur nächsten. Der Probenrobot sollte daher als gesamte Einheit in einem abgeschlossenen System mit konstanter Luftfeuchtigkeit und geschützt vor möglichen Luftbewegungen stehen. Auch zur Minimierung von Kontaminationen erscheint ein solcher Aufbau empfehlenswert.

4.3 Kompatibilität der Probenträgerformate zwischen Pipettierrobot und MALDI-MS

Pipettiersystem und MALDI-Massenspektrometer arbeiten im gegebenen Fall nicht mit 100% kompatiblen Formaten. Das MultiPROBE II System ist, wie auch alle anderen, auf dem Markt gängigen Pipettiersysteme für Flüssigkeiten auf den Gebrauch von 96er und 384er Mikrotiterplatten ausgelegt. Das durch die SCOUT 26 Ionenquelle des Bruker Reflex III vorgegebene Targetformat (rund, mit irregulärer, radialer Spotanordnung) hat sich zwar für die Flüssigkeitsabgabe in der Methode *MALDI-LS 1.42s* an das Mikrotiterplattenformat adaptieren lassen, ist jedoch alles andere als optimal. Eine mittlerweile ebenfalls erhältliche Ionenquelle SCOUT MTP besitzt 384er Mikrotiterplattenformat und ist damit 1:1 kompatibel mit den Vorgaben durch das Pipettiersystem. Der Programmablauf der Methode *MALDI-LS 1.42s* ließe sich unter Verwendung der entsprechend kompatiblen MALDI-Targets an mehreren Stellen deutlich vereinfachen. Zusätzlich wäre eine wesentlich schnellere Präparation möglich und auch der häufige Targetwechsel bei der automatischen MALSI-MS Messung verringert sich.

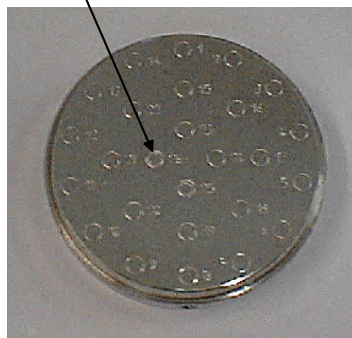
4.4 Automatische MALDI-MS Messungen

Auch hinsichtlich der automatischen MALDI-MS Messungen sind Verbesserungen anzustreben, die vorrangig eine einfache und schnellere Messung der DHB-Präparationen erlauben. Eine Optimierung der Messbedingungen ist in diesem Zusammenhang ebenfalls eng an eine apparative Veränderung der Ionenquelle des MALDI-Massenspektrometers gebunden, was nachfolgend kurz erläutert werden soll.

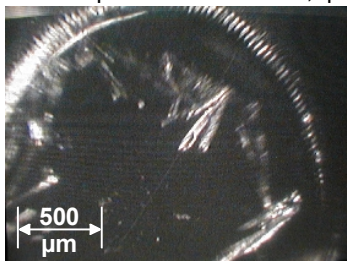
Eine prinzipielle Verbesserung der Morphologie der DHB-Präparationen für die automatische MALDI-MS Messung lässt sich über die Verwendung von MALDI-Targets erreichen, die auf der sogenannten „Anker Chip“-Technologie basieren. Diese Targets sind mit einer hydrophoben, teflonähnlichen Schicht überzogen, die auf den Spotpositionen hydrophile Ankergruppen besitzen. Die Ankergruppen sind mit 200-800µm Durchmesser wesentlich kleiner als die übliche Spotgröße (2mm). Beim Eintrocknen der wässrigen Lösungen erfolgt einerseits eine Konzentrierung der Probe, andererseits sind die Matrixkristalle in ihrer Position auf dem Spot besser definiert. Abbildung 87 zeigt einen Vergleich des Aussehens der Probenpräparationen auf dem Standard SCOUT 26 Target und den Anker-Targets bei Auftragsvolumina von 0,5µl und 1,0µl. Alle Abbildungen besitzen den gleichen Maßstab.

Abbildung 87: Vergleich der Probenpräparationen mit Standard SCOUT 26 Target und Anker-Targets

Standard SCOUT 26 Target:
Spotdurchmesser 2mm
(umrandete Stellen)



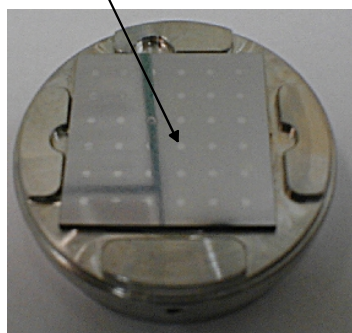
2mm Spotdurchmesser - 0,5µl



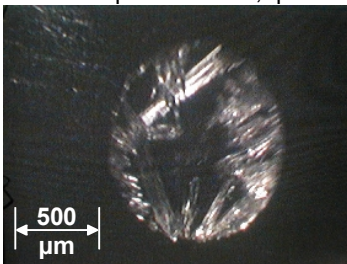
2mm Spotdurchmesser - 1,0µl



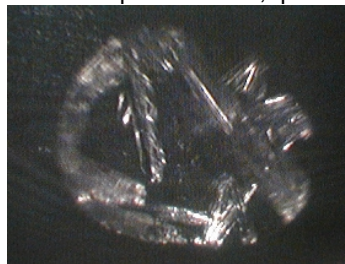
SCOUT 26 Anker-Target:
Spotdurchmesser 800-200µm
(helle Punkte - hier gezeigt
für 800µm Spotdurchmesser)



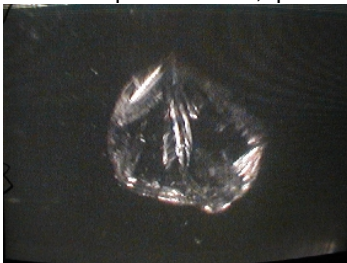
800µm Anker - 0,5µl



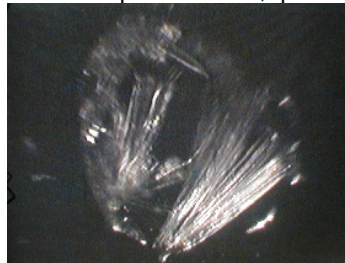
800µm Anker - 1,0µl



600µm Anker - 0,5µl



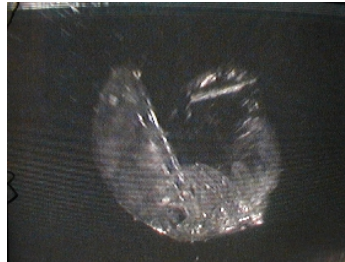
600µm Anker - 1,0µl



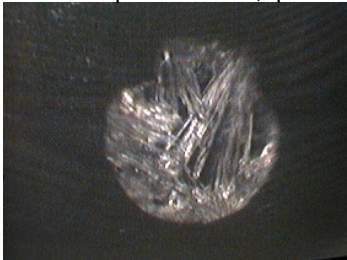
400µm Anker - 0,5µl



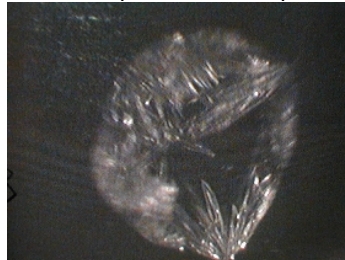
400µm Anker - 1,0µl



200µm Anker - 0,5µl

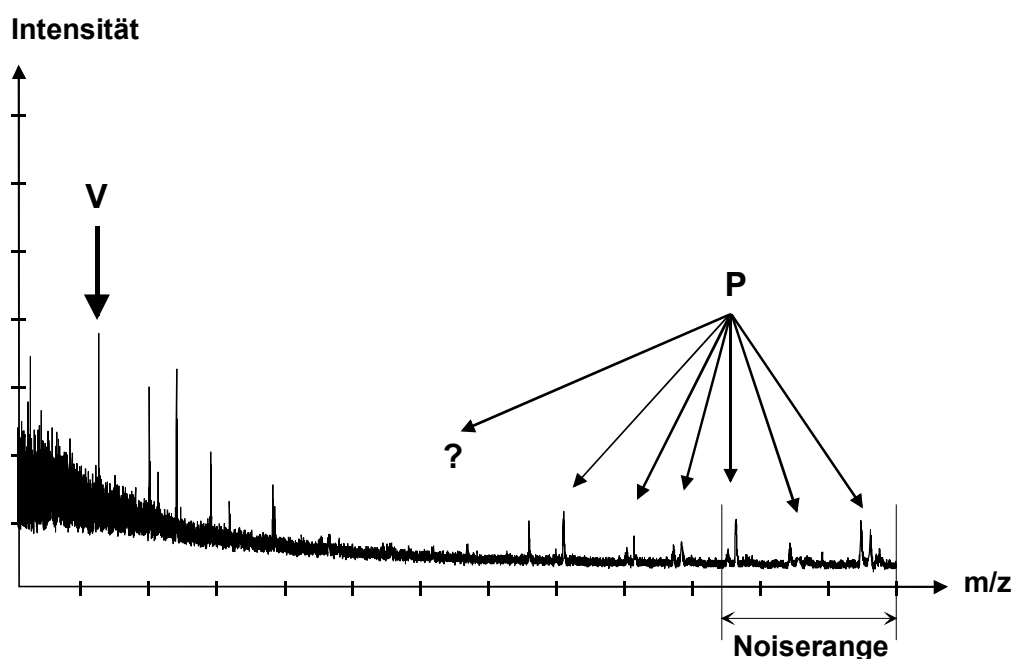


200µm Anker - 1,0µl



Der störende Hystereseeffekt beim Anfahren der Spotpositionen führt allerdings dazu, dass in Verbindung mit der SCOUT 26 Ionenquelle die Anker-Targets für eine automatische Messung nicht geeignet sind. Erst in Verbindung mit der SCOUT MTP Ionenquelle ist eine sinnvolle Anwendung dieser Technik möglich. Bei exaktem Anfahren der Probenspots durch die Automatik sollten sich dann DHB-Präparationen in gleicher Weise für die automatische Messung eignen, wie CHCA-Präparationen, mit dem zusätzlichen Vorteil einer stärkeren Konzentrierung der Probe und somit einer gesteigerten Empfindlichkeit bei der Messung. Zusätzliche nötige Verbesserungen bei der automatischen MALDI-MS Messung betreffen die Softwarekomponente *AutoXecute* und hier speziell die Auswahl geeigneter Einzelspektren bei der Summierung. Bei den Versuchen erwies sich die Berechnung des Signal/Rausch-Verhältnisses durch die AutoXecute als kritischer Punkt. Üblicherweise wird dieses Verhältnis über das zum Signal gehörende Rauschen berechnet. Bei der automatischen Messung durch die AutoXecute wird jedoch zur Berechnung des Rauschens allgemein der obere m/z -Bereich des Spektrums herangezogen wie dies in Abbildung 88 gezeigt ist. („Noiserange“)

Abbildung 88: Kriterien zur Berechnung des Signal/Rausch-Verhältnisses



MALDI-MS Spektren zeigen jedoch typischerweise vom Bereich niedriger Massen zum Bereich höherer Massen einen abklingenden Verlauf des Rauschens, analog dem in Abbildung 88 gezeigten Spektrum. In der Regel liegt der maximale Ausschlag des Rauschens bei ca. 500 Da um eine Größenordnung über dem maximalen Ausschlag des Rauschens bei ca. 3000 Da. Daher werden Peaks, wie z.B. der angedeutete Peak einer Verunreinigung oder

eines Salzes (V) bezüglich ihres Signal/Rauschverhältnisses gegenüber den eigentlich interessierenden Signalen der Leiterpeptide (angedeutet durch „P“) überbewertet. Für die Summe wurden damit häufig Einzelspektren berücksichtigt, die in ihrer Intensität den Schwerpunkt im unteren Massenbereich des Spektrums haben. Eine analytisch korrekte Berechnung des Signal/Rausch-Verhältnisses innerhalb der Automatik-Software stellt eine wichtige Forderung für eine Verbesserung der Methode dar.

Eine weitere Verbesserung ließe sich durch die Definition von einzelnen Massen oder gar Massenbereichen erzielen, die für die Spektrenbeurteilung im Hinblick auf die Summation ignoriert werden. Häufig treten nämlich bei einer Messreihe die Massen bestimmte Verunreinigungen oder Salze immer wieder auf. Die *Fuzzy Logic* berücksichtigt aber bei der Optimierung der Messparameter und der Spektrenaddition den gesamten Messbereich und optimiert somit eventuell sogar auf die Signale dieser Verunreinigungen oder Salze und berücksichtigt diese als Kriterium für die Spektrenaddition. Das Problem einer fehlerhaften Spektrenaddition ließe sich also auch dadurch minimieren, dass Massenbereiche, in denen solche Signale auftreten, von einer Bewertung durch die *Fuzzy Logic* ausgeschlossen werden können.

4.5 Aussagekraft negativer Resultate

Unter Berücksichtigung der genannten Probleme die bei der automatischen Messung von DHB-Präparationen auftreten können, erscheint eine Überprüfung negativer Resultate nach einem Automatiklauf meist notwendig. Wird ein Spot bei der Messung „aufgegeben“ oder zeigt das aufgenommene Spektrum nur Signale die keine eindeutige Interpretation zulassen, so ist es ratsam, die Messung der Probe manuell zu überprüfen. Erst dadurch kann ausgeschlossen werden, dass das negative Resultat lediglich durch den Laserbeschuss ungeeigneter Stellen auf der Präparation zustande kam.

Kapitel IV:

Zusammenfassung und Ausblick

Ziel dieser Arbeit sollte die Entwicklung einer automatisierten, hochparallelen und nachweisstarken Sequenzierungsmethode für Proteine im Rahmen der Proteomanalytik sein. Zur Konzeption der Methodik sollte dabei mit der sogenannten Leitersequenzierung eine Kopplung aus enzymatischem Aminosäureabbau und MALDI-Massenspektrometrie zum Einsatz kommen. Die experimentellen Grundparameter für die Leitersequenzierung im Bezug auf die nasschemischen Sequenzierungsschritte und die massenspektrometrischen Messung wurden dabei im ersten Teil der Arbeit zunächst manuell ausgearbeitet. Im zweiten Abschnitt wurden dann die Möglichkeiten einer Automatisierung der so erarbeiteten Methode zur Leitersequenzierung auf verschiedenen Ebenen evaluiert.

Da der enzymatische Abbau eines Gesamtproteins durch dessen intrinsische Eigenschaften wie Sekundärstruktur, Tertiärstruktur (Zugänglichkeit der Proteintermini für die Exopeptidase) oder Lösungsverhalten erschwert ist, wurden die Sequenzierungsprotokolle der Leitersequenzierung für proteolytisch erzeugte, RP-HPLC-gereinigte Spaltfragmente von Proteinen optimiert. Zudem besitzt die MALDI-MS, wie auch andere gängige massenspektrometrische Verfahren, im Massenbereich über ca. 5-10 kDa eine zu geringe Auflösung und Genauigkeit, um eine eindeutige Zuordnung der Massendifferenzen in den Leiterpeptidspektren zu den durch Exopeptidase abgespaltenen Aminosäure zu erlauben. Bei Versuchen mit verschiedenen Endopeptidasen stellte sich – entgegen theoretischen Erwartungen – heraus, dass vor allem auch Proteinspaltungen mit Endoproteinase GluC (im Phosphatpuffer) und Chymotrypsin, neben Spaltungen mit Endoproteinase LysC, gute Voraussetzungen für die nachfolgende enzymatische Sequenzierung liefern. Mit den Proteinfragmenten aus Spaltungen der Endoproteinasen GluC (im Phosphatpuffer) und Chymotrypsin wurden bei den enzymatischen Sequenzierungen die höchsten Sequenzabdeckungen erzielt.

In der technischen Ausführung wurden alle Experimente im Hinblick auf eine Sequenzierung direkt auf dem Target (*on-target*) optimiert. Geringfügige Anpassungen der erarbeiteten Methoden erlaubten jedoch auch eine Sequenzierung von Peptiden, die zuvor auf PVDF-Membran (Immobilon PSQ) aufgetragen wurden. Ein relativ kontrollierter und reproduzierbarer Abbau war in einem Temperaturbereich von ca. 25-35°C möglich. Die enzymatischen Sequenzierungen erfolgten ausschließlich mit kommerziell erhältlichen Exopeptidasepräparationen. Eine zusätzliche Aufreinigung der eingesetzten Enzyme erwies sich als nicht notwendig. Um eine einfache Interpretation der massenspektrometrischen Resultate zu gewährleisten, wurden nur Mono-peptidylpeptidasen eingesetzt. Für C-terminale Sequenzierungen wurden mit CPY, CPP, CPW, CPA und CPB Carboxypeptidasen unterschiedlicher Spezifität einzeln und in verschiedenen Kombinationen untersucht. Von der N-terminal Seite aus wurden

Sequenzierungen mit APM, LAP, API und AAP durchgeführt. Eine Betrachtung der Sequenzierungen zeigt dabei, dass zumeist nur aus kombinierten Resultaten der Anwendung verschiedener Carboxy- beziehungsweise Aminopeptidasen eine ausreichende Sequenzinformation erhalten werden kann.

C-terminal ist der Anteil sequenzliefernder Fragmente bei der Verwendung von CPY, sowie bei den sequenziellen Peptidasenkombinationen sb(CP-I) und den Peptidasenmischungen pb(CP) mit maximal 42% am höchsten. Hier treten auch vermehrt längere Teilsequenzen mit bis zu 12 AS auf. N-terminal hebt sich APM als Einzelpeptidase mit überdurchschnittlich guten Sequenzresultaten gegenüber den anderen Aminopeptidasen ab. Bis zu 34% der getesteten Peptide liefern allein schon mit diesem Enzym eine Sequenzinformation. Zudem wurden längere Teilsequenzen mit bis zu 9 AS im Vergleich zu anderen Aminopeptidasen häufiger erhalten. Sequenzierungen mit Aminopeptidasenmischungen bei N-terminaler Sequenzierung brachten im Gegensatz zu Carboxypeptidasenmischungen bei C-terminaler Sequenzierung kaum deutliche Vorteile. Auch sehr lange Sequenzabschnitte mit bis zu 16 AS N-terminal und bis zu 25 AS C-terminal wurden in Einzelfällen erhalten. C- und N-terminal am häufigsten wurden jedoch aus den Leiterspektren Teilsequenzen mit bis zu 6 AS erhalten (85% aller sequenzliefernden Proteinfragmente). Der größte Teil der Sequenzabdeckung stammt mit 70% aus Teilsequenzen mit 3-9 AS.

Unter dem Hauptaspekt einer möglichen Steigerung der Sequenzinformation bei den Leitersequenzierung wurden unterschiedliche Peptidderivatisierungen untersucht. Die gezielte N-terminale Modifizierungen brachte in vielen Fällen durch die resultierende Massenverschiebung des derivatisierten Peptids einen Gewinn an C-terminaler Sequenzinformation. Gegenüber entsprechend nicht-modifizierten Peptiden konnten durch die Massenverschiebung auch Leiterpeptide beobachtet werden, die sonst mit Signalen der MALDI-Matrix interferieren. Bei Modifikationsreagenzien, die eine fixierte positive Ladung oder ein ausgedehntes, delokalisiertes Elektronensystem ins Peptid einbrachten, wie z.B. Sulforhodamin B oder FMOC-NHS wurde dabei zusätzlich auch eine sehr gute Response im Massenspektrum beobachtet. Die Empfindlichkeiten lagen selbst bei der Sequenzierung der Rohprodukte solcher Derivate im unteren fmol-Bereich. Das charakteristische Isotopenmuster Brom-haltiger Peptidderivate erlaubte weiterhin ein stark vereinfachtes Auslesen der Leitersequenz aus dem Massenspektrum. Während die Leitersequenzierung allgemein keine Unterscheidung der isobaren Aminosäuren Leucin und Isoleucin erlaubt, gelang die Unterscheidung der ebenfalls isobaren Aminosäuren Lysin und Glutamin nach einer schnellen und selektiven Acetylierung des Lysins mit anschließender C-terminaler Sequenzierung.

Im Hinblick auf die Analyse post-translationaler Modifikationen wurden insbesondere Phosphorylierungen eingehender untersucht. Dabei war sowohl bei der N-terminalen, als auch bei der C-terminalen Leitersequenzierung ein enzymatischer Abbau der phosphorylierten Aminosäuren zu beobachten und damit die entsprechende Phosphorylierungsstelle schnell und eindeutig zu identifizieren. Die N-terminale Sequenzierung mit APM lieferte die besten Resultate und ermöglichte sowohl den Abbau von Phosphotyrosin wie auch Phosphoserin, während der C-terminale Abbau sich auf Phosphotyrosin beschränkt zeigte.

Aus der Summe der erhaltenen Resultate (Sequenzen) folgt, dass die Leitersequenzierung unter den gegebenen Voraussetzungen – insbesondere der limitierenden Verfügbarkeit zusätzlicher Exopeptidasen mit ergänzenden Spaltungsspezifitäten – im wesentlichen als Instrument zur schnellen Generierung kurzer Sequenztags geeignet ist. Auf diesem Gebiet stellt die erarbeitete Leitersequenzierung im Vergleich zur Edman-Sequenzierung eine wesentlich schnellere Alternative dar. Im Gegensatz zu rein massenspektrometrischen Sequenzierungen ist die Interpretation der Sequenzen im Massenspektrum stark vereinfacht und daher zumeist eindeutig. Die Empfindlichkeit der Methode ist stark von der untersuchten Peptidsequenz abhängig. Generelle Werte für die Empfindlichkeit lassen sich somit nicht angeben, die Resultate lassen jedoch für eine Peptidausgangsmenge im oberen fmol-Bereich (1000-500fmol) eine Leitersequenzierung ausnahmslos möglich erscheinen. Die Ergebnisse von Verdünnungsreihen zeigen zudem, dass auch nach Sequenzierung von Peptidmengen im mittleren bis unteren fmol-Bereich (50-10fmol) ein Auslesen der Peptidsequenz aus dem Massenspektrum zumeist möglich ist. Zum Erreichen solcher Empfindlichkeiten ist die weitgehende Minimierung von Suppressionseffekten und damit die Verwendung von DHB als MALDI-Matrix eine notwendige Voraussetzung.

Eine Evaluierungsstudie mit vier verschiedenen Proteinen führt zum Schluss, dass die Methodik auch für die *de novo* Sequenzierung unbekannter Proteine ein hohes Potential birgt. Die ermittelte Sequenzabdeckung der überlappenden Spaltfragmente lag bei maximal 80%. Im Bereich prolinhaltiger Sequenzabschnitte fehlen Überlappungen dabei am häufigsten, da auf Seiten der N-terminalen Sequenzierung geeignete Exopeptidasen zur Spaltung der Iminbindung nicht verfügbar waren. Zur Herstellung von Oligonucleotidsequenzen, die dann als Hybridisierungs sonden in der Nucleinsäureanalytik eingesetzt werden, ist die Länge der erhaltenen Sequenzabschnitte jedoch in fast allen Fällen ausreichend.

Die Methoden „MALDI-LS 1.42s“ für die Probenpräparation mit dem Pipettierroboter MultiPROBE II, sowie „multiprobe_5“ für die MALDI-MS Messung mittels AutoXecute auf dem Bruker Reflex III Massenspektrometer, erlauben eine Umsetzung der manuell ausgearbeiteten Methodik auf ein vollständig automatisiertes System.

Eine Excel-Tabellenvorlage, die in konvertierter Form von beiden beteiligten Geräten gelesen werden kann, ermöglicht eine zentrale und einfache Dateneingabe für die Proben. Diese Art der Dateneingabe erlaubt im Zusammenspiel mit dem ausgearbeitetem Automatisierungsprogramm eine vollkommen flexible, individuelle Behandlung der einzelnen Proben. Die erforderliche „Ortspräzision“ bei der Abgabe der Flüssigkeiten auf dem MALDI-Target konnte durch eine entsprechenden ausgelegte Performance-Datei für den Pipettiervorgang erreicht werden. Querkontaminationen beim Pipettieren wurden durch organische Spülschritte in der Methode eliminiert. In der Summe lieferten die auf dem automatischen Pipettiersystem mit der Methode *MALDI-LS 1.42s* präparierten Proben im Massenspektrum vergleichbare Abbauspektren und somit auch die gleiche Sequenzinformation, wie entsprechend manuell präparierte Proben.

Bei den automatischen MALDI-MS Messungen war eine Anpassung der Parameter insbesondere aufgrund der bevorzugten Verwendung der DHB-Matrix notwendig. Ein erhöhter Acetonitrilgehalt von 50% im Lösungsmittel der DHB sorgte für eine Verbesserung der Präparation im Bezug auf die automatische *AutoXecute* Messung. Mit speziellen Rasterkoordinaten, die bei der Laserabtastung der einzelnen Probenspots auf die besonderen Kristallisationseigenschaften der DHB-Matrix zugeschnitten wurden, konnten gute Ergebnisse erzielt werden. Vorteile zeigten die DHB-Präparationen, im Vergleich zu CHCA-Präparationen, in der Toleranz gegenüber geringfügigen Mengen von Puffersalzen, wie sie bei der enzymatischen Sequenzierung üblich sind. Die Toleranzschwelle für den Abbruch der automatischen Messung musste jedoch bei den enzymatisch sequenzierten Proben und Verwendung von DHB-Matrix im Vergleich zu Messungen salzfreier Peptidproben in CHCA-Matrix erhöht werden, was im Durchschnitt zu etwas längeren Messzeiten führte.

Die in dieser Arbeit weiterentwickelten Methoden zur enzymatischen *on-target* Sequenzierung von Peptiden erlauben somit, in Verbindung mit den beschriebenen Hard- und Softwarekomponenten zur automatischen Probenpräparation und MALDI-MS Messung, deren Einsatz für eine schnelle Sequenzierung in der Proteinanalytik.

Zusätzliche Verbesserungen könnten jedoch, bei entsprechender Verfügbarkeit, noch durch Exopeptidasen mit ergänzender Spaltungsspezifität (z.B. X-Pro Aminopeptidase, EC 3.4.11.9) erzielt werden. Auf Seiten der Automatisierung ergäben sich durch die ausschließliche Verwendung eines 96er oder 384er Mikrotiterplattenformat auf allen Geräten (Pipettierrobot und MALDI-MS) deutliche Vereinfachungen in der Methode, bei gleichzeitig noch höherem Probendurchsatz.

Anhang

A. Abkürzungsverzeichnis

ACN	Acetonitril
amol	Attomol
AS	Aminosäure(n)
ASA	Aminosäureanalyse
BS	Bernsteinsäure
bp	Basenpaar(e)
CBZ	Carbobezoxy (Benzyloxycarbonyl)
CHCA	α -Cyano-4-hydroxyzimtsäure
CNBr	Bromcyan
Da	Dalton
DAD	Diodenarray Detektor
DE	delayed ion extraction
DHB	2,5-Dihydroxybenzoesäure
DHBs	super-DHB
<i>DrTI</i>	Serin Proteinase Inhibitor (<i>Delonix regia Trypsin Inhibitor</i>)
DTT	1,4-Dithioerythrit
EDTA	Ethylendiamintetraessigsäure
ESI-MS	Elektrospray-Ionisation
ff.	fortfolgende
fmol	Femtomol
FWHM	Halbwertsbreite (Breite des Peaks auf halber Höhe; <i>full width at half maximum</i>)
HMBS	2-Hydroxy-5-methoxybenzoesäure
HPLC	Hochleistungs-Flüssigchromatographie (<i>high performance liquid chromatography</i>)
HPTLC	Hochleistungs-Dünnschichtchromatographie
IR	Infrarot
kDa	Kilodalton
LS	Leitersequenzierung (<i>ladder sequencing</i>)
MALDI	Matrix-unterstützte Laserdesorptions/Ionisation

MCP	<i>microchannel plates</i>
MSL	<i>Multiprobe Script Language</i>
μl	Mikroliter
mM	Millimolar
ms	Millisekunde(n)
MS	Massenspektrometrie
MS/MS	Tandem-Massenspektrometrie
mU	Milliunit
NaCit	Natriumcitrat
nmol	Nanomol
OT-LS	<i>on-target</i> Leitersequenzierung (<i>on-target ladder sequencing</i>)
PC	Personal Computer
PCR	Ploymerase-Kettenreaktion (<i>polymerase chain reaction</i>)
PIE	pulsed ion extraction
pK _a	Säurekonstante
pmol	Pikomol
PP	Polypropylen
PSD	<i>post source decay</i>
PVDF	Polyvinylidenfluorid
R _n	(restliche) Peptidkette mit beliebigen Aminosäuren
RP-HPLC	<i>reversed-phase</i> Hochleistungs-Flüssigchromatographie
S.	Seite
SA	3,5-Dimethoxy-4-hydroxyzimtsäure (Sinapinsäure)
TCEP	Tris(2-carboxyethyl)-phosphin
Temp.	Temperatur
TFA	Trifluoressigsäure
TOF	<i>time of flight</i>
Tris	Tris(hydroxymethyl)-aminomethan
UV	Ultraviolett
YAG	Yttrium-Aluminium-Granat
Xaa, Xbb, Xcc	Aminosäuren

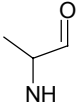
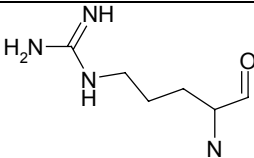
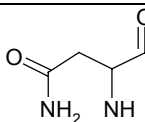
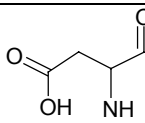
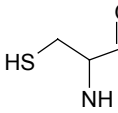
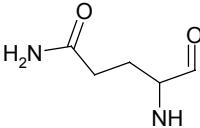
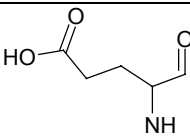
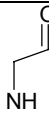
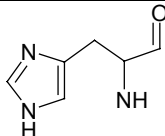
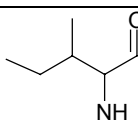
B. Nomenklatur der Peptidmodifikationen

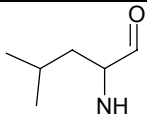
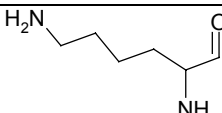
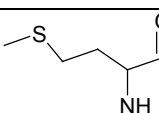
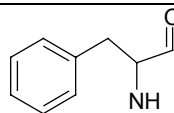
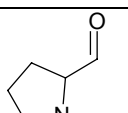
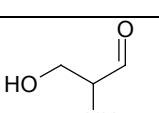
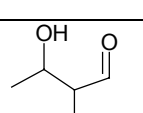
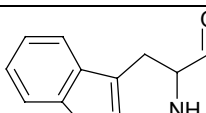
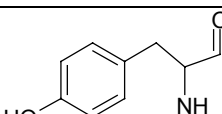
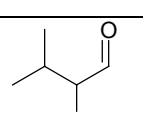
(pyro)E	Pyroglutaminsäure (auch Pyr oder pGlu)
Ac	Acetylgruppe
phos	Phosphatgruppe
PE	Pyridylethyl
BB	Brombenzoyl
BH	Bromhexanoyl
Fmoc	Fluorenmethoxycarbonyl
BBOC	Brombenzyloxycarbonyl
RB	Rhodamin B
ACQ TM	6-Aminochinolylcarbamoyl
E5	Eosin 5
SRB	Sulforhodamin B
DNS	Dansyl
DABS	Dabsyl

C. Abkürzungen der Peptidasen

CPA	Carboxypeptidase A
CPB	Carboxypeptidase B
CPC	Carboxypeptidase C
CPP	Carboxypeptidase P (<i>Penicillium janthinellum</i>)
CPW	Carboxypeptidase W (<i>wheat</i>)
CPY	Carboxypeptidase Y (<i>yeast</i>)
APM	(Leucin) Aminopeptidase microsomal
AAP	Aminopeptidase <i>aeromonas proteolytica</i>
API	Aminopeptidase I
LAP	Leucin Aminopeptidase cytosolisch
LysC	Endoproteinase LysC
GluC	Endoproteinase GluC
GluC (Carbonat)	Endoproteinase GluC (angewendet in Carbonatpuffer)
GluC (Phosphat)	Endoproteinase GluC (angewendet in Phosphatpuffer)
AspN	Endoproteinase AspN
ArgC	Endoproteinase ArgC
CTR	Chymotrypsin
TR	Trypsin

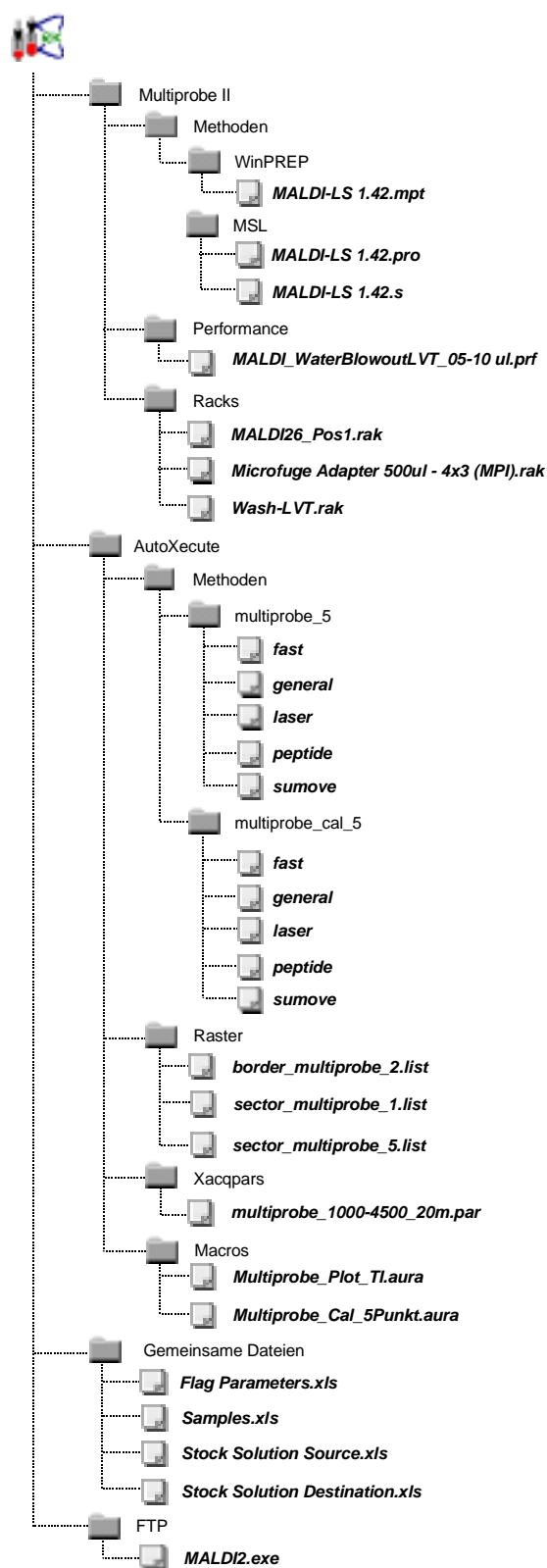
D. Aminosäuren / Aminosäurereste

Bezeichnung	Code		Struktur des Aminosäurerests in der Peptidkette - Bindung zur nächsten AS nicht eingezeichnet	beobachtete Massendifferenz im Leiter- spektrum bei Abspaltung:	
				monoisotopische Masse	Durchschnittsmasse
Alanyl Alanin	Ala	A		71,03711	71,0788
Arginyl Arginin	Arg	R		156,10111	156,1876
Asparaginyll Asparagin	Asn	N		114,04293	114,1039
α-Aspartyl Asparaginsäure	Asp	D		115,02694	115,0886
Cysteinyl Cystein	Cys	C		103,00919	103,1448
Glutaminyl Glutamin	Gln	Q		129,04259	129,1155
α-Glutamyl Glutaminsäure	Glu	E		128,05858	128,1308
Glycyl Glycin	Gly	G		57,02146	57,0520
Histidyl Histidin	His	H		137,05891	137,1412
Isoleucyl Isoleucin	Ile	I		113,08406	113,1595

Bezeichnung	Code		Struktur des Aminosäurerest in der Peptidkette - Bindung zur nächsten AS nicht eingezeichnet	beobachtete Massendifferenz im Leiter- spektrum bei Abspaltung:	
				monoisotopische Masse	Durchschnittsmasse
Leucyl Leucin	Leu	L		113,08406	113,1595
Lysyl Lysin	Lys	K		128,09496	128,1742
Methionyl Methionin	Met	M		131,04049	131,1986
Phenylalanyl Phenylalanin	Phe	F		147,06841	147,1766
Prolyl Prolin	Pro	P		97,05276	97,1167
Seryl Serin	Ser	S		87,03203	87,0782
Threonyl Threonin	Thr	T		101,04768	101,1051
Tryptophanyl Tryptophan	Trp	W		186,07931	186,2133
Tyrosyl Tyrosin	Tyr	Y		163,06333	163,1760
Valyl Valin	Val	V		99,06841	99,1326

Nomenklatur und Symbole der Aminosäuren entsprechen den Empfehlungen der INTERNATIONAL UNION OF PURE AND APPLIED CHEMISTRY (IUPAC) und der INTERNATIONAL UNION OF BIOCHEMISTRY AND MOLECULAR BIOLOGY von 1983 [Moss_2, 2000].

E. Verzeichnis der Dateien auf beiliegender CD-ROM



Auf der linken Seite ist die Verzeichnisstruktur der im Text besprochenen Dateien auf der beiliegenden CD-ROM gezeigt. Alle Dateien sind sowohl im Originalformat des jeweiligen Programms, wie auch als simple Textdateien im ASCII-Format auf dieser CD-ROM abgelegt.

Handhabung der CD-ROM:

1) PC:

Beim Einlegen ins CD-ROM Laufwerk startet die CD automatisch ein Menü, über das die jeweiligen Dateien kopiert, aber auch betrachtet werden können. Alternativ können alle Dateien auch ohne dieses Menü mit jedem beliebigen Texteditor betrachtet werden. Für diesen Fall sind die Dateien unter dem Verzeichnis „Originaldateien und Textformat für PC und MAC“ bei gleicher Verzeichnisstruktur leichter zu finden.

Anmerkung zur pdf-Version der Arbeit:

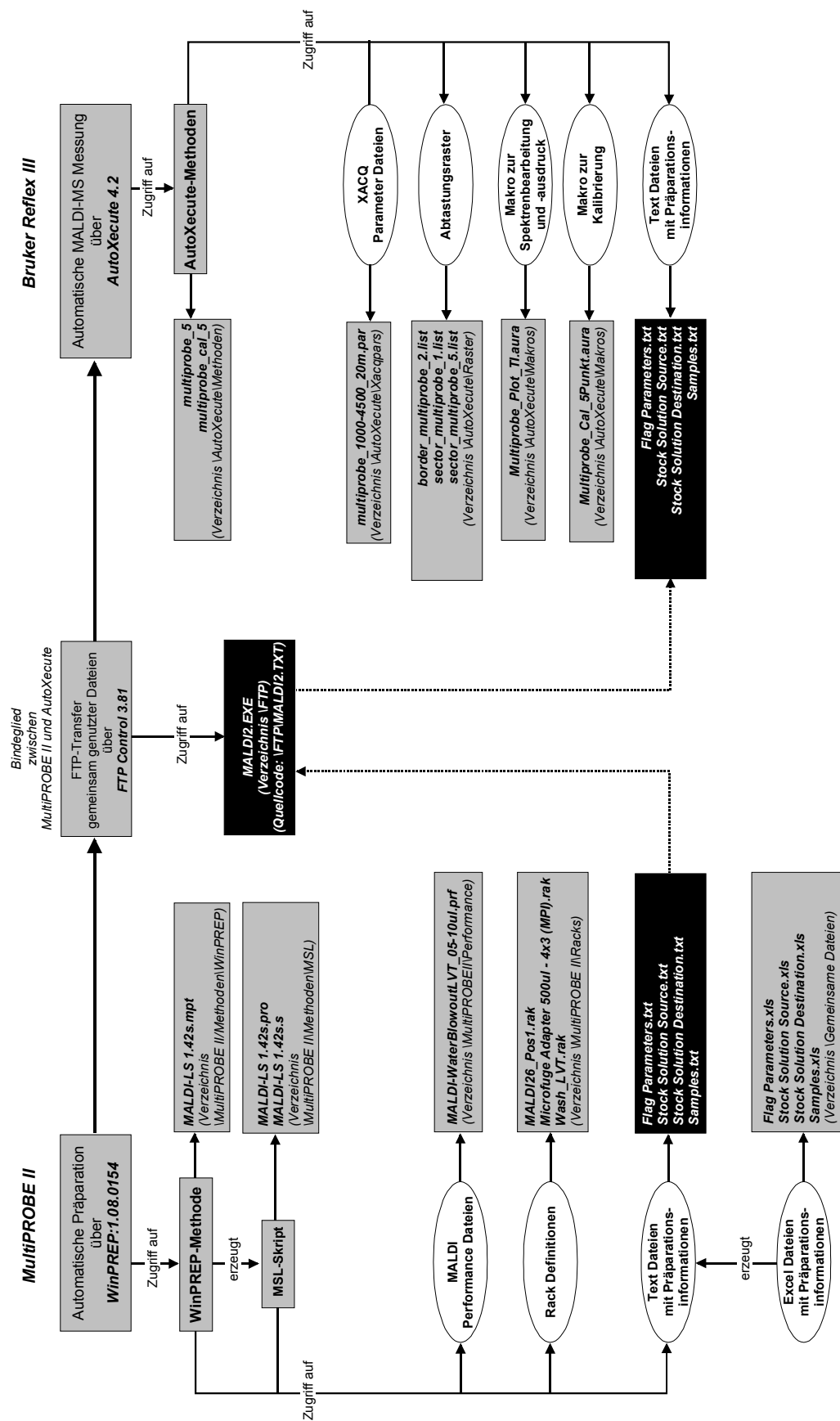
Das Menü muß manuell über die Datei „cdm.exe“ gestartet werden

2) Apple Macintosh:

Alle genannten Dateien sind unter dem Verzeichnis „Originaldateien und Textformat für PC und MAC“ bei gleicher Verzeichnisstruktur wie in der Darstellung links zu finden. Die Dateien im ASCII-Format können mit jedem beliebigen Texteditor geöffnet werden.

Überblick über einige wichtige Dateien für die Ausführung der Automatisierung

(Alle angegebenen Verzeichnisse beziehen sich auf die beiliegende CD zur Arbeit, nicht auf die Originalverzeichnisse in den jeweiligen Programmen !)



F. Literaturverzeichnis

- Alvarez-Coque**, M. C. G., Hernandez, M. J. M., Camanas, R. M. V., Fernandez, C. M., *Formation and Instability of o-Phthalaldehyde Derivatives of Amino Acids*, Anal. Biochem. 178 (1989) 1-7
- Ambler**, R., *Carboxypeptidases A and B*, Methods Enzymol. 25B (1972) 262-272
- Cohen**, S., De Antonis, K. Application of Amino Acid Derivatization with 6-Aminoquinolyl-N-hydroxysuccinimidyl Carbamate, J. Chom. A 661 (1994) 25-34
- De Antonis**, K., Brown, P., Cheng, Y-F., Cohen, S., Analysis of Derivatized Peptides by Capillary Electrophoresis, J. Chom. A 661 (1994) 279-285
- Barlet-Jones**, M., Jeffery, W., Hansen, H., Pappin, D., *Peptide Ladder Sequencing by Mass Spectrometry Using a Novel, Volatile Degradation Reagent*, Rapid. Commun. Mass. Spectrom. 8 (1994) 773-742
- Biemann**, K., Martin, S.A., *Mass Spectrometric Determination of the Amino Acid Sequences of Peptides and Proteins*, Mass Spectrom. Rev., 1987, 6, 1-76
- Bonetto**, V., Bergman A.-C., Jörnvall, H., Sillard, R., *C-Terminnal Sequence Analysis of Peptides and Proteins Using Carboxypeptidases and Mass Spectrometry after Derivatization of Lys and Cys Residues*, Anal. Chem. 69 (1997) 1315-1319
- Boyd**, V., Bozzini, M., Zon, G., Noble, R., Mattaliano, R., *Sequencing of Peptides and Proteins from the Carboxy Terminus*, Anal. Biochem. 206 (1992) 344-352
- Brown**, R., Lennon, J., *Mass Resolution Improvement by Incorporation of Pulsed Ion Extraction in a Matrix-Assisted Laser Desorption/Ionization Linear Time-of-Flight Mass Spectrometer*, Anal. Chem. 67 (1995) 1998-2003
- Callahan**, P., McDonald, K., Ellis, S., *Sequencing of Peptides With Dipeptidyl Aminopeptidase I*, Methods Enzymol., 25B (1972) 282-298
- Caprioli**, R., Fan, T., *Peptide Sequence Analysis Using Exopeptidases With Molecular Analysis of the Truncated Polypeptides by Mass Spectrometry*, Anal. Biochem. 154 (1986) 596-603
- Chait**, B. Wang, R. Beavis, R., Kent, S., *Protein Ladder Sequencing*, Science 262 (1993) 89-92
- Chen**, G., Edwards, T., D'souza, V., Holz, R., *Mechanistic Studies on the Aminopeptidase form Aeromonas Proteolytica: A Two-Metal Ion Mechanism for Peptide Hydrolysis*, Biochemistry 36 (1997) 4278-4286
- Cohen**, S., Chait, B., *Influence of Matrix Solution Conditions on the MALDI-MS Analysis of Peptides and Proteins*, Anal. Chem. 68 (1996) 31-37
- Davis**, J., Giddings, J., *Statistical Theory of Component Overlap in Multicomponent Chromatograms*, Anal. Chem. 55 (1983) 418
- Doolittle**, R. F. in Fasman, G. D. (Ed.), *Predictions of Protein Structure and the Principles of Protein Conformation*, Plenum Press (1989)
- Doucette**, A., Li, L., *Protein Identification by Mass Fingerprinting Using Trypsin Followed by Partial N-terminal Sequencing With Leucine Aminopeptidase M*, 47th Conference of the American Society for Mass Spectrometry, Dallas Texas (1999)

- Ducka**, D., Cornett, L., Kräuter, K.O., *Automatic acquisition of MALDI-TOF mass spectra*, *Analisis Magazine* 26 (1998) M36-M40
- Ducret**, A., Van Oostveen, I., Eng, J., Yates III, J., Aebersold R., *High Throughput Protein Characterization by Automated Reverse-Phase Chromatography/Electrospray Tandem Mass Spectrometry*, *Protein Science*. 7 (1998) 706-719
- Eckerskorn**, C., Strupat, K., Kellermann, J., Lottspeich, F., Hillenkamp, F., *High-Sensitivity Peptide Mapping by Micro-LC With On-Line Membrane Blotting and Subsequent Detection by Scanning-IR-MALDI Mass Spectrometry*, *J. Protein Chem.* 16 (1997) 349-362
- Edman**, P., *Acta Chem. Scand.* 4 (1950) 283-290
- Einarsson**, S., Josefsson, B., Lagerkvist, S., *Determination of Amino Acids With 9-Fluorenmethyl Chloroformate and Reversed-Phase High-Performance Liquid Chromatography*, *J. Chromatogr.* 282 (1983) 609-618
- Gross**, J., Strupat, K., *Matrix-Assisted Laser Desorption/Ionisation-Mass Spectrometry Applied to Biological Macromolecules*, *Trends in Analytical Chemistry* 17 (1998) 470-484
- Gu**, Q-M., Prestwich, G., *Efficient Peptide Ladder Sequencing by MALDI-TOF Mass Spectrometry Using Allyl Isothiocyanate*, *J. Peptide Res.* 49 (1997) 484-491
- Haberhausen**, G., *Polymerase-Kettenreaktion in Bioanalytik*, Eds.: F. Lottspeich, H. Zorbas, Spektrum Verlag, Heidelberg (1998) 673-703
- Hanson**, H., *Crystalline Leucin Aminopeptidase from Lens (α-Aminoacyl-Peptide Hydrolase; EC 3.4.11.1)*, *Methods Enzymol.*, 45 (1976) 504-521
- Hayashi**, R., *Carboxypeptidase Y in Sequence Determination of Peptides*, *Methods Enzymol.* 47 (1977) 84-93
- Hayashi**, R., *Carboxypeptidase Y*, *Methods Enzymol.* 45 (1976) 568-587
- Hayashi**, R., Hata, T., *Biochim. Biophys. Acta*, 263 (1972) 673
- Heinrikson**, R. L., Meridith, S. C., *Amino Acid Analysis by Reversed-Phase High-Performance Liquid Chromatography: Precolumn Derivatization With Phenylisothiocyanate*, *Anal. Biochem.* 136 (1984) 65-74
- Holland**, R., Heinze, T., Lay, J., *Leucine Aminopeptidase Digestion and MALDI TOF/MS Analysis for Partial Sequencing of Single and Double Chain Peptides*, *Proceedings of the 45th American Society for Mass Spectrometry Conference on Mass Spectrometry and Allied Topics*, Palm Beach, California, 1997
- Huheey**, J., *Anorganische Chemie: Prinzipien von Struktur und Reaktivität*, Walter de Gruyter, Berlin (1988) 309-354
- Jensen**, O., Larsen, M., Roepstroff, P., *Mass Spectrometric Identification and Microcharacterization of Proteins From Electrophoretic Gels: Strategies and Applications*, *Proteins: Structure, Function and Genetics – Suppl. 2*, 1998, 74-89
- Jensen**, O., Vorm, O., Mann, M., *Sequence Patterns Produced by Incomplete Enzymatic Digestion or One-Step Edman Degradation of Peptide Mixtures as Probes for Protein Database Searches*, *Electrophoresis* 17 (1996) 938-944

- Karas, M., Bahr, U., Gießmann, U.,** *Matrix-Assisted Laser Desorption Ionization Mass Spectrometry*, Mass Spectrometry Reviews 10 (1991) 335-357
- Karas, M., Hillenkamp, F.,** *Laser Desorption Ionization of Proteins With Molecular Masses Exceeding 10000 Daltons*, Anal. Chem. 60 (1988) 2299-2301
- Karlin, S., Altschul, S.F.,** *Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes*, Proc. Natl. Acad. Sci. USA 87 (1990) 2264-2268
- Karlskov, K. Breddam, K., Roepstroff, P.,** *C-Terminal Sequence Determination of Peptides Degraded with Carboxypeptidases of Different Specificities and Analyzed by 252-Cf Plasma Desorption Mass Spectrometry*, Anal. Biochem. 180 (1989) 28-37
- Kaufmann, R., Spengler, B., Lützenkirchen, F.,** *Mass Spectrometric Sequencing of Peptides by Product-ion Analysis in a Reflectron Time-of-flight Mass Spectrometer Using Matrix-assisted Laser Desorption Ionization*, Rapid Commun. Mass Spectrom. 7 (1993) 902-9100
- Keough, T., Youngquist, R., Lacey, M.,** *A Method for High-Sensitivity Peptide Sequencing Using Postsource Decay Matrix-Assisted Laser Desorption Ionization Mass Spectrometry*, Proc. Natl. Acad. Sci. USA 96 (1999) 7131-7136
- Kessler, C.,** *Hybridisierung und Nachweistechiken in Bioanalytik*, Eds.: F. Lottspeich, H. Zorbas, Spektrum Verlag, Heidelberg (1998) 635-671
- Kratzer_1, R., Eckerskorn, C., Karas, M., Lottspeich, F.,** *Suppression Effects in Enzymatic Peptide Ladder Sequencing Using Ultraviolet-Matrix Assisted Laser Desorption/Ionization-Mass Spectrometry*, Electrophoresis 19 (1998) 1910-1919
- Kratzer_2, R., Eckerskorn, C., Karas, M., Lottspeich, F.,** *Suppression Effects in UV- and IR-MALDI-MS*, Proceedings of the 46th American Society for Mass Spectrometry Conference on Mass Spectrometry and Allied Topics, Orlando, Florida, 1998
- Kratzer, R., Lottspeich, F.,** *Improvement in Enzymatic Peptide Ladder Sequencing Coupled With UV-MALDI-TOF-MS Using Peptide Derivatization*, Proceedings of the 47th American Society for Mass Spectrometry Conference on Mass Spectrometry and Allied Topics, Dallas, Texas, 1999
- Liao, P.-C., Allison, J.,** *Enhanced Detection of Peptides in Matrix-Assisted Laser Desorption/Ionization Mass Spectrometry Through Use of Charge-Localized Derivatives*, J. Mass Spectrom. 30 (1995) 511
- Light, A.,** *Leucin Aminopeptidase in Sequence Determination of Peptides*, Methods Enzymol. 25B (1972) 253-262
- Lin, J. K., Chang, J. Y.,** *Chromophoric Labeling of Amino Acids With 4-Dimethylamino-azobenzene-4-sulphonyl Chloride*, Anal. Chem. 47 (1975) 1634-1638
- Lottspeich, F.,** *Proteom Analysis: A Pathway to the Functional Analysis of Proteins*, Angew. Chem. Int. Ed. 38 (1999) 2476-2492
- Mamyrin, B., Karataev, V., Shmikk, D., Zagulin, V.,** Sov. Phys. JETP, 37 (1973) 45-48
- Mann, M., Hojrup, P., Roepstroff, P.,** *Use of Mass Spectrometric Molecular Weight Information to Identify Proteins in Sequence Databases*, Biol. Mass Spectrom. 22 (1993) 338-345

- Mann, M.**, *Sequence Database Searching by Mass Spectrometric Data* in Microcharacterization of Proteins, Eds.: R. Kellner, F. Lottspeich, H. Meyer, Wiley-VCH, Weinheim (1994) 223-245
- Manning, J. M.**, *The Contributions of Stein and Moore to Protein Science*, Protein Science 2 (1993) 188-191
- Martin, M.**, Herman, D., Guichon, G., Anal. Chem. 58 (1986) 2200
- Mewes, H.-W.**, George, D., *Protein Sequences and Protein Databases* in Microcharacterization of Proteins, Eds.: R. Kellner, F. Lottspeich, H. Meyer, Wiley-VCH, Weinheim (1994) 209-222
- Meyer, H.**, *Analyzing Post-translational Protein Modifications* in Microcharacterization of Proteins, Eds.: R. Kellner, F. Lottspeich, H. Meyer, Wiley-VCH, Weinheim (1994) 131-146
- Moore, S.** Stein, W. H., *Photometric Ninhydrin Method for Use in the Chromatography of Amino Acids*, J. Biol. Chem., 176 (1948) 367-388
- Moss_1, G.**, *Enzyme Nomenclature from the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB)* Supplement 6 (06/2000), World Wide Web Version at the Department of Chemistry, Queen Mary and Westfield College, <http://www.chem.qmw.ac.uk/uibmb/enzyme/>
- Moss_2, G.**, *Nomenclature and Symbolism for Amino Acids and Peptides – Recommendations of the International Union of Pure and Applied Chemistry (IUPAC) and the International Union of Biochemistry and Molecular Biology (IUBMB) from 1983*, World Wide Web Version at the Department of Chemistry, Queen Mary and Westfield College, <http://www.chem.qmw.ac.uk/iupac/aminoacid/>
- Olumee, Z.**, Sadeghi, M., Tang, X., Vertes, A., *Amino Acid Composition and Wavelength Effects in Matrix-Assisted Laser Desorption/Ionization*, Paid Commun. Mass Spectrom. 9 (1995) 744-752
- Patterson, D.**, Tarr, G., Regnier, F., Martin, S., *C-terminal Ladder Sequencing via Matrix-Assisted Laser Desorption Mass Spectrometry Coupled With Carboxypeptidase Y Time-Dependent and Concentration-Dependent Digestions*, Anal. Chem. 67 (1995) 3971-3978
- Perkins, D.**, Pappin, D. Creasy, D., Cottrell, J., *Probability-based Protein Identification by Searching Sequence Databases Using Mass Spectrometry Data*, Electrophoresis, 20 (1999), 3551-3567
- Peterson, L.**, Sokolovsky, M., Vallee, B., Biochemistry, 15 (1976) 2501-2508
- Prescott, J.**, Wilkes, S., *Aeromonas Aminopeptidase*, Methods Enzymol. 25B (1972) 530-541
- Schechter, I.**, *On the Active Sites of Proteases: Cleavage of Peptide Bonds Involving D-Alanine Residues by Carboxypeptidase A*, Eur. J. Biochem., 14 (1970) 526-520
- Schlack, P.**, Kumpf, W., *Über eine neue Methode zur Ermittlung der Konstitution von Peptiden*, Z. Physiol. Chem. 154 (1926) 125-170
- Schomburg, D.**, *BRENDA, the Enzyme Database* (06/2000) World Wide Web Version des Instituts für Biochemie der Universität zu Köln <http://www.brenda.uni-koeln.de/>
- Schreiner, M.**, Strupat, K., Lottspeich, F., Eckerskorn, C., *Ultraviolet matrix assisted laser desorption ionization-mass spectrometry of electroblotted Proteins*, Electrophoresis 17 (1996) 954-961

- Seidl, R.**, *Peptidsequenzierung mit Dipeptidyl-Aminopeptidasen und Tandem-Massenspektrometrie*, Dissertation an der Ludwig-Maximilians-Universität München (1989)
- Shen, T.-L.**, Allison, J., *Interpretation of Matrix-Assisted Laser Desorption/Ionization Post Source Decay Spectra of Charge-Derivatized Peptides: Some Examples of Tris[(2,4,6-Trimethoxyphenyl)-Phosphonium]-Tagged Proteolytic Products of Phosphoenolpyruvat Carboxykinase*, J.Am. Soc. Mass Spectrom. 11 (2000) 145-152
- Shevchenko, An.**, Wilm, M., Vorm, O., Jensen, O., Podtelejenikov, A., Neubauer, G., Shevchenko, Al., Mortensen, P., Mann, M., Biochem. Mass Spectrom., Biochem. Soc. Trans. 24 (1996) 893-896
- Spengler, B.**, Kirsch, D., Kaufmann, R., Jaeger, E., *Peptide Sequencing by Matrix-assisted Laser-desorption Mass Spectrometry*, Rapid Commun. Mass Spectrom. 6 (1992) 105-108
- Spengler, B.**, Lützenkirchen, F., Metzger, S., Chaurand, P., Kaufmann, R., Jeffery, W., Barlet-Jones, M., Pappin, D., *Peptide Sequencing of Charged Derivatives by Postsource Decay MALDI Mass Spectrometry*, Int. J. Mass Spectrom. Ion Proc. 169/170 (1997) 127-140
- Spröbler, B.**, Heilmann, H.-D., Grampp, E., Uhlig, H., *Eigenschaften der Carboxypeptidase C aus Orangenblättern*, Hoppe-Seyler's Z. Physiol. Chem.. 352 (1971) 1524-1530
- Suckau, D.**, Cornett, L., Kräuter, K.O., *Automatic acquisition of MALDI-TOF mass spectra*, Analisis Magazine, 26 (1998) M36-M40
- Süßmuth, R.** Jung, G., *Impact of Mass Spectrometry on Combinatorial Chemistry*, J. Chrom. B, 1999, 725, 49-65
- Takamoto, K.**, Kamo, M., Kubota, K., Satake, K., Tsugita, A., *Carboxy-Terminal Degradation of Peptides Using Perfluoracyl Anhydrides: A C-terminal Sequencing Method*, Eur. J. Biochem. 228 (1995) 362-372
- Tapuhi, Y.**, Schmidt, D. E., Lindner, W., Karger, B. L., *Dansylation of Amino Acids for High Performance Liquid Chromatography Analysis*, Anal. Biochem. 115 (1981) 123-129
- Taylor, A.**, Aminopeptidases: towards a mechanism of action, TIBS 18 (1993), 167-172
- Thiede, B.**, Salnikow, J., Wittmann-Liebold, B., *C-terminal Ladder Sequencing by an Approach Combining Chemical Degradation With Analysis by Matrix-Assisted-Laser-Desorption Ionization Mass Spectrometry*, Eur. J. Biochem. 244 (1997) 750-754
- Thiede, B.**, Wittmann-Liebold, B., Bienert, M., Krause, E., *MALDI-MS for C-terminal Sequence Determination of Peptides and Proteins Degraded by Carboxypeptidase Y and P*, FEBS Letters 357 (1995) 65-69
- Tschesche, H.**, *Carboxypeptidase C*, Methods Enzymol. 47 (1977) 73-82
- Tsugita, A.**, Takamoto, K., Kamo, M., Iwadata, H., *C-terminal Sequencing of Protein: A Novel Partial Acid Hydrolysis and Analysis by Mass Spectrometry*, Eur. J. Biochem. 206 (1992) 691-696
- Umetsu, H.**, Hishinuma, K., Wake, H., Takeuchi, M., Ichishima, E., *Action of Carboxypeptidase W on Oligopeptides Containing Carboxy-Terminally Amidated Peptides*, Phytochemistry 5 (1997) 907-910
- Vestal, M.**, Juhasz, P., Martin, S., *Delayed Extraction Matrix-assisted Laser Desorption Time-of-Flight Mass Spectrometry*, Rapid. Commun. Mass. Spectrom. 9 (1995) 1004-1050

- Vestling**, M., Fenselau, C., *Poly(vinylidene difluoride) Membranes as the Interface between Laser Desorption Mass Spectrometry, Gel Electrophoresis, and in Situ Proteolysis*, Anal. Chem., 66 (1994) 471-477
- Voet**, D., Voet, J., *Biochemistry 2nd. Edition*, John Wiley & Sons, New York (1995) 345-370
- Yokoyama**, S., Obayashi, A., Tanabe, D., Ichishima, E., Biochim. Biophys. Acta, 397 (1975) 443-448
- Wiley**, W., McLaren, I., Rev. Sci. Instrum. 26 (1953) 1150-1157
- Wilkins**, M., Sanchez, J.-C., Gooley, A., Appel, R., Humphery-Smith, I., Hochstrasser, D., Williams, K., Biotechnology and Genetic Engineering Reviews 13 (1995) 19-50
- Wilkins_1**, M., Pasquali C., Appel, R., Ou, K., Gonzales, O., Sanchez, J., Yan, J., Gooley, A., Hughes, G., Humpherysmith, I., Williams, K., Hochstrasser, D., *From Proteins to Proteomes – Large Scale Protein Identification by Two-Dimensional Electrophoresis and Amino Acid Analysis*, Bio-Technology 14 (1996) 61-65
- Wilkins_2**, M., Gasteiger, E., Sanchez, Appel, R., J.-C., Hochstrasser, D., *Protein Identification with Sequence Tags*, Current Biology 6 (1996) 1543-1544
- Yates**, J., *Mass Spectrometry from Genomic to Proteomics*, Trends in Genetics, 16 (2000) 5-8
- Yates**, J., McCormack, A., Link, A., Schieltz, D., Eng, J., Hays, L., *Future Prospects for the Analysis of Complex Biological Systems Using Micro-column Liquid Chromatography-Electrospray Tandem Mass Spectrometry*, The Analyst, 121 (1996) 65R-76R
- Zenobi**, R., Knochenmuss, R., *Ion Formation in MALDI Mass Spectrometry*, Mass Spectrometry Reviews 17 (1998) 337-366
- Zuber**, H., Carboxypeptidase C, Methods Enzymol., 45 (1976) 561-568

Getrennter Anhang

(gemäß §12 Abs. 6 der Promotionsordnung vom 29. Januar 1998)

der

Dissertation

zur Erlangung des Doktorgrades

der Fakultät für Chemie und Pharmazie

der Ludwig-Maximilians-Universität München



Entwicklung einer *High-Throughput*-Sequenzierungsmethode für die Proteomanalytik

Robert Georg Kratzer

aus

Augsburg

München, März 2001

Inhaltsverzeichnis

-

getrennter Anhang

1	Die Datei "MALDI-LS 1_42s.pro"	1
2	Die Datei "MALDI-LS 1_42s.s"	2
3	Die Datei "MALDI2.exe – Quellcode"	170

1 Die Datei “MALDI-LS 1_42s.pro”

c:\biochem\packard\multiprobe\bin\WinPREP.s

C:\BIOCHEM\Packard\MultiPROBE\bin\Robert\Method Packages\MALDI-LS 1.4s\MALDI-LS 1.41s\MALDI-LS 1.42s.s

2 Die Datei "MALDI-LS 1_42s.s"

```

/*****
*
*   MultiPROBE II WinPREP Test Script
*   -----
*
*   Based On:      C:\BIOCHEM\Packard\MultiPROBE\bin\Robert\Method Packages\MALDI-LS 1.4s\MALDI-LS
1.41s\MALDI-LS 1.42s.MPT
*   Generated By:   WinPREP Version 1.08.0154
*   Date Generated: 11/23/00 14:34:37
*   Username:      Administrator
*   System:        MultiPROBE
*
*****/

// The name and path of this test.
char *pszTestName = "MALDI-LS 1";
char *pszTestPath = "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-LS 1.4s\\MALDI-
LS 1.41s";

/*****
*
*   Runtime Variable Definitions
*
*****/

char   Rt_ExternalDataFileName[260];
char   Rt_Calibrant[260];
char   Rt_GeneralTestInformation[260];
char   Rt_StockSolSource[260];
char   Rt_StockSolDest[260];
double Rt_dDspVolume_S1 = 30;
double Rt_dDspVolume_S2 = 30;
double Rt_dDspVolume_S3 = 30;
MP2_TIME Rt_stTimePeriod_S1;
MP2_TIME Rt_stTimePeriod_S2;
double Rt_dDspVolume_S4 = 30;
double Rt_dDspVolume_S5 = 30;
double Rt_dDspVolume_S6 = 30;
double Rt_dDspVolume_S7 = 30;
double Rt_dDspVolume_S8 = 30;
double Rt_dDspVolume_S9 = 30;
double Rt_dDspVolume_S10 = 20;

int IniAllRtVariables()
{
    strcpy( Rt_ExternalDataFileName, "Sample Data File" );
    strcpy( Rt_Calibrant, "Calibrant File" );
    strcpy( Rt_GeneralTestInformation, "General Test Information" );
    strcpy( Rt_StockSolSource, "Stock Solution Source" );
    strcpy( Rt_StockSolDest, "Stock Solution Destination" );
    Rt_stTimePeriod_S1.bPeriod = 1;
    MSL_StringToTime( "0 00:10:00", &Rt_stTimePeriod_S1 );
    Rt_stTimePeriod_S2.bPeriod = 1;
    MSL_StringToTime( "0 00:05:00", &Rt_stTimePeriod_S2 );

    return 0;
}

/*****
*
*   Runtime File Definitions
*
*****/

char* NULL = 0; // Global NULL pointer value

```

```
MP2_RTFILE_DEF RtFile1;
MP2_RTFILE_DEF RtFile2;
MP2_RTFILE_DEF RtFile3;
MP2_RTFILE_DEF RtFile4;
MP2_RTFILE_DEF RtFile5;
MP2_RTFILE_DEF RtFile6;
MP2_RTFILE_DEF RtFile7;
MP2_RTFILE_DEF RtFile8;
MP2_RTFILE_DEF RtFile9;
MP2_RTFILE_DEF RtFile10;
MP2_RTFILE_DEF RtFile11;
MP2_RTFILE_DEF RtFile12;
MP2_RTFILE_DEF RtFile13;
MP2_RTFILE_DEF RtFile14;
MP2_RTFILE_DEF RtFile15;
MP2_RTFILE_DEF RtFile16;
MP2_RTFILE_DEF RtFile17;
MP2_RTFILE_DEF RtFile18;
MP2_RTFILE_DEF RtFile19;
MP2_RTFILE_DEF RtFile20;
MP2_RTFILE_DEF RtFile21;
MP2_RTFILE_DEF RtFile22;
MP2_RTFILE_DEF RtFile23;
MP2_RTFILE_DEF RtFile24;
MP2_RTFILE_DEF RtFile25;
MP2_RTFILE_DEF RtFile26;
MP2_RTFILE_DEF RtFile27;
MP2_RTFILE_DEF RtFile28;
MP2_RTFILE_DEF RtFile29;
MP2_RTFILE_DEF RtFile30;
MP2_RTFILE_DEF RtFile31;
MP2_RTFILE_DEF RtFile32;
MP2_RTFILE_DEF RtFile33;
MP2_RTFILE_DEF RtFile34;
MP2_RTFILE_DEF RtFile35;
MP2_RTFILE_DEF RtFile36;
MP2_RTFILE_DEF RtFile37;
MP2_RTFILE_DEF RtFile38;
MP2_RTFILE_DEF RtFile39;
MP2_RTFILE_DEF RtFile40;
MP2_RTFILE_DEF RtFile41;
MP2_RTFILE_DEF RtFile42;
MP2_RTFILE_DEF RtFile43;
MP2_RTFILE_DEF RtFile44;
MP2_RTFILE_DEF RtFile45;
MP2_RTFILE_DEF RtFile46;
MP2_RTFILE_DEF RtFile47;
MP2_RTFILE_DEF RtFile48;
MP2_RTFILE_DEF RtFile49;
MP2_RTFILE_DEF RtFile50;
MP2_RTFILE_DEF RtFile51;
MP2_RTFILE_DEF RtFile52;
MP2_RTFILE_DEF RtFile53;
MP2_RTFILE_DEF RtFile54;
MP2_RTFILE_DEF RtFile55;
MP2_RTFILE_DEF RtFile56;
MP2_RTFILE_DEF RtFile57;
MP2_RTFILE_DEF RtFile58;
MP2_RTFILE_DEF RtFile59;
MP2_RTFILE_DEF RtFile60;
MP2_RTFILE_DEF RtFile61;
MP2_RTFILE_DEF RtFile62;
MP2_RTFILE_DEF RtFile63;
MP2_RTFILE_DEF RtFile64;
MP2_RTFILE_DEF RtFile65;
MP2_RTFILE_DEF RtFile66;
MP2_RTFILE_DEF RtFile67;
MP2_RTFILE_DEF RtFile68;
MP2_RTFILE_DEF RtFile69;
MP2_RTFILE_DEF RtFile70;
MP2_RTFILE_DEF RtFile71;
MP2_RTFILE_DEF RtFile72;
MP2_RTFILE_DEF RtFile73;
MP2_RTFILE_DEF RtFile74;
MP2_RTFILE_DEF RtFile75;
MP2_RTFILE_DEF RtFile76;
MP2_RTFILE_DEF RtFile77;
MP2_RTFILE_DEF RtFile78;
```

```
MP2_RTFILE_DEF RtFile79;
MP2_RTFILE_DEF RtFile80;
MP2_RTFILE_DEF RtFile81;
MP2_RTFILE_DEF RtFile82;
MP2_RTFILE_DEF RtFile83;
MP2_RTFILE_DEF RtFile84;
MP2_RTFILE_DEF RtFile85;
MP2_RTFILE_DEF RtFile86;
MP2_RTFILE_DEF RtFile87;
MP2_RTFILE_DEF RtFile88;
MP2_RTFILE_DEF RtFile89;
MP2_RTFILE_DEF RtFile90;
MP2_RTFILE_DEF RtFile91;
MP2_RTFILE_DEF RtFile92;
MP2_RTFILE_DEF RtFile93;
MP2_RTFILE_DEF RtFile94;
MP2_RTFILE_DEF RtFile95;
MP2_RTFILE_DEF RtFile96;
MP2_RTFILE_DEF RtFile97;
MP2_RTFILE_DEF RtFile98;
MP2_RTFILE_DEF RtFile99;
MP2_RTFILE_DEF RtFile100;
MP2_RTFILE_DEF RtFile101;
MP2_RTFILE_DEF RtFile102;
MP2_RTFILE_DEF RtFile103;
MP2_RTFILE_DEF RtFile104;
MP2_RTFILE_DEF RtFile105;
MP2_RTFILE_DEF RtFile106;
MP2_RTFILE_DEF RtFile107;
MP2_RTFILE_DEF RtFile108;
MP2_RTFILE_DEF RtFile109;
MP2_RTFILE_DEF RtFile110;
MP2_RTFILE_DEF RtFile111;
MP2_RTFILE_DEF RtFile112;
MP2_RTFILE_DEF RtFile113;
MP2_RTFILE_DEF RtFile114;
MP2_RTFILE_DEF RtFile115;
MP2_RTFILE_DEF RtFile116;
MP2_RTFILE_DEF RtFile117;
MP2_RTFILE_DEF RtFile118;
MP2_RTFILE_DEF RtFile119;
MP2_RTFILE_DEF RtFile120;
MP2_RTFILE_DEF RtFile121;
MP2_RTFILE_DEF RtFile122;
MP2_RTFILE_DEF RtFile123;
MP2_RTFILE_DEF RtFile124;
MP2_RTFILE_DEF RtFile125;
MP2_RTFILE_DEF RtFile126;
MP2_RTFILE_DEF RtFile127;
MP2_RTFILE_DEF RtFile128;
MP2_RTFILE_DEF RtFile129;
MP2_RTFILE_DEF RtFile130;
MP2_RTFILE_DEF RtFile131;
MP2_RTFILE_DEF RtFile132;
MP2_RTFILE_DEF RtFile133;
MP2_RTFILE_DEF RtFile134;
MP2_RTFILE_DEF RtFile135;
MP2_RTFILE_DEF RtFile136;
MP2_RTFILE_DEF RtFile137;

int IniAllRtFileDefs()
{
    MSL_IniRtFileDef( &RtFile1, Rt_ExternalDataFileName, 1, NULL, 0, ' ', 4, sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile2, Rt_Calibrant, 1, NULL, 0, ' ', 3, sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile3, Rt_GeneralTestInformation, 1, NULL, 0, ' ', 3, sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile4, Rt_StockSolSource, 1, NULL, 0, ' ', 3, sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile5, Rt_StockSolDest, 1, NULL, 0, ' ', 3, sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile6, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-LS
1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M0C04_Aspirate_OrganicWash_LVT.csv", 0, NULL, 0, ',', 1,
sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile7, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-LS
1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M0C05_Dispende_OrganicWash_LVT.csv", 0, NULL, 0, ',', 1,
sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile8, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-LS
1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M0C06_AqueousFlush_after_OrganicWash_LVT.csv", 0, NULL, 0, ',', 1,
sizeof(MP2_RTFILE_DEF) );
}
```

[illegible]

[illegible]

[illegible]

[illegible]

```

    MSL_IniRtFileDef( &RtFile122, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M050301_Flush_1.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF)
);
    MSL_IniRtFileDef( &RtFile123, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M050302_Wash_1.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile124, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M050401_Flush_1.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF)
);
    MSL_IniRtFileDef( &RtFile125, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M050402_Wash_1.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile126, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M1D02_AqueousFlush.csv", 0, NULL, 0, ',', 1,
sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile127, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M1D03_AqueousWash.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF)
);
    MSL_IniRtFileDef( &RtFile128, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M1E02_AqueousFlush.csv", 0, NULL, 0, ',', 1,
sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile129, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M1E03_AqueousWash.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF)
);
    MSL_IniRtFileDef( &RtFile130, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M2302_AqueousFlush.csv", 0, NULL, 0, ',', 1,
sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile131, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M2303_AqueousWash.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF)
);
    MSL_IniRtFileDef( &RtFile132, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M2402_AqueousFlush.csv", 0, NULL, 0, ',', 1,
sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile133, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M2403_AqueousWash.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF)
);
    MSL_IniRtFileDef( &RtFile134, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M2902_AqueousFlush.csv", 0, NULL, 0, ',', 1,
sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile135, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M2903_AqueousWash.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF)
);
    MSL_IniRtFileDef( &RtFile136, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M2A02_AqueousFlush.csv", 0, NULL, 0, ',', 1,
sizeof(MP2_RTFILE_DEF) );
    MSL_IniRtFileDef( &RtFile137, "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-
LS 1.4s\\MALDI-LS 1.41s\\MALDI-LS 1.42s_M2A03_AqueousWash.csv", 0, NULL, 0, ',', 1, sizeof(MP2_RTFILE_DEF)
);

    return 0;
}

/*****
*
*   Sample List Definitions
*
*****/

int IniAllRtSmpListDefs()
{
    return 0;
}

/*****
*
*   Well Map Definitions
*
*****/

MP2_WELLMAP_DEF M0C00_Aspirate_Samples;
MP2_WELLMAP_DEF M0C01_Disperse_Samples;
MP2_WELLMAP_DEF M0C04_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M0C05_Disperse_OrganicWash_LVT;
MP2_WELLMAP_DEF M0C06_AqueousFlush_after_OrganicWash_LVT;

```

```
MP2_WELLMAP_DEF M0C07_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M1B00_Aspirate_Enzyme_1;
MP2_WELLMAP_DEF M1B01_Dispense_Enzyme_1;
MP2_WELLMAP_DEF M1B02_AqueousFlush;
MP2_WELLMAP_DEF M1B03_AqueousWash;
MP2_WELLMAP_DEF M0200_Pre_Test_Flush_1;
MP2_WELLMAP_DEF M0201_Pre_Test_Wash_1;
MP2_WELLMAP_DEF M2D04_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M2D05_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M2D00_Aspirate_DHB_Matrix_for_Samples;
MP2_WELLMAP_DEF M2D01_Dispense_DHB_Matrix_for_Samples;
MP2_WELLMAP_DEF M2D06_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M2D07_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M0E00_Aspirate_Calibrant;
MP2_WELLMAP_DEF M0E01_Dispense_Calibrant;
MP2_WELLMAP_DEF M0E06_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M0E07_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3400_Aspirate_Matrix_for_Calibrant;
MP2_WELLMAP_DEF M3401_Dispense_Matrix_for_Calibrant;
MP2_WELLMAP_DEF M3404_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M3405_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M3406_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3407_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M0E04_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M0E05_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M3700_Aspirate_Matrix_for_Calibrant_optional;
MP2_WELLMAP_DEF M3701_Dispense_Matrix_for_Calibrant_optional;
MP2_WELLMAP_DEF M3704_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M3705_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M3706_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3707_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3000_Aspirate_Matrix_for_Samples_optional;
MP2_WELLMAP_DEF M3001_Dispense_Matrix_for_Samples_optional;
MP2_WELLMAP_DEF M3004_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M3005_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M3006_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3007_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M0C02_AqueousFlush_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M0C03_AqueousWash_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M0E02_AqueousFlush_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M0E03_AqueousWash_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M2D02_AqueousFlush_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M2D03_AqueousWash_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M3003_AqueousWash_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M3002_AqueousFlush_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M2100_Aspirate_Enzyme_2;
MP2_WELLMAP_DEF M2101_Dispense_Enzyme_2;
MP2_WELLMAP_DEF M2102_AqueousFlush;
MP2_WELLMAP_DEF M2103_AqueousWash;
MP2_WELLMAP_DEF M2700_Aspirate_Enzyme_3;
MP2_WELLMAP_DEF M2701_Dispense_Enzyme_3;
MP2_WELLMAP_DEF M2702_AqueousFlush;
MP2_WELLMAP_DEF M2703_AqueousWash;
MP2_WELLMAP_DEF M050101_Flush_1;
MP2_WELLMAP_DEF M050102_Wash_1;
MP2_WELLMAP_DEF M05010000_Aspirate;
MP2_WELLMAP_DEF M05010001_Dsp1_Dispense;
MP2_WELLMAP_DEF M0801_Flush_1;
MP2_WELLMAP_DEF M0802_Wash_1;
MP2_WELLMAP_DEF M080000_Aspirate;
MP2_WELLMAP_DEF M080001_Dsp1_Dispense;
MP2_WELLMAP_DEF M030001_Flush_1;
MP2_WELLMAP_DEF M030002_Wash_1;
MP2_WELLMAP_DEF M03000000_Aspirate;
MP2_WELLMAP_DEF M03000001_Enzyme_1_Dispense;
MP2_WELLMAP_DEF M030101_Flush_1;
MP2_WELLMAP_DEF M030102_Wash_1;
MP2_WELLMAP_DEF M03010000_Aspirate;
MP2_WELLMAP_DEF M03010001_Dsp1_Dispense;
MP2_WELLMAP_DEF M030201_Flush_1;
MP2_WELLMAP_DEF M030202_Wash_1;
MP2_WELLMAP_DEF M03020000_Aspirate;
MP2_WELLMAP_DEF M03020001_Dsp1_Dispense;
MP2_WELLMAP_DEF M050201_Flush_1;
MP2_WELLMAP_DEF M050202_Wash_1;
MP2_WELLMAP_DEF M05020000_Aspirate;
MP2_WELLMAP_DEF M05020001_Dsp1_Dispense;
MP2_WELLMAP_DEF M050000_Flush_1;
```

```
MP2_WELLMAP_DEF M050001_Wash_1;
MP2_WELLMAP_DEF M1100_Aspirate_Buffer_1;
MP2_WELLMAP_DEF M1101_Dispense_Buffer_1;
MP2_WELLMAP_DEF M1102_AqueousFlush_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M1103_AqueousWash_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M1104_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M1105_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M1106_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M1107_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M1400_Aspirate_Reagent;
MP2_WELLMAP_DEF M1401_Dispense_Reagent;
MP2_WELLMAP_DEF M1402_AqueousFlush_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M1403_AqueousWash_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M1404_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M1405_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M1406_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M1407_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M1700_Aspirate_Buffer_2;
MP2_WELLMAP_DEF M1701_Dispense_Buffer_2;
MP2_WELLMAP_DEF M1702_AqueousFlush_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M1703_AqueousWash_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M1704_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M1705_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M1706_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M1707_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M0A00_Flush_1;
MP2_WELLMAP_DEF M0A01_Wash_1;
MP2_WELLMAP_DEF M3402_AqueousFlush_before_OrganicWash_LVT_1;
MP2_WELLMAP_DEF M3403_AqueousWash_before_OrganicWash_LVT_1;
MP2_WELLMAP_DEF M3703_AqueousWash_before_OrganicWash_LVT_2;
MP2_WELLMAP_DEF M3702_AqueousFlush_before_OrganicWash_LVT_2;
MP2_WELLMAP_DEF M070002_Flush_1;
MP2_WELLMAP_DEF M070003_Wash_1;
MP2_WELLMAP_DEF M070000_Aspirate;
MP2_WELLMAP_DEF M070001_Dispense;
MP2_WELLMAP_DEF M090002_Flush_1;
MP2_WELLMAP_DEF M090003_Wash_1;
MP2_WELLMAP_DEF M090000_Aspirate;
MP2_WELLMAP_DEF M090001_Dispense;
MP2_WELLMAP_DEF M2E00_Aspirate_alpha_Matrix_for_Samples;
MP2_WELLMAP_DEF M2E01_Dispense_alpha_Matrix_for_Samples;
MP2_WELLMAP_DEF M2E02_AqueousFlush_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M2E03_AqueousWash_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M2E04_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M2E05_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M2E06_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M2E07_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3100_Aspirate_alpha_Matrix_for_Samples;
MP2_WELLMAP_DEF M3101_Dispense_alpha_Matrix_for_Samples;
MP2_WELLMAP_DEF M3102_AqueousFlush_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M3103_AqueousWash_before_OrganicWash_LVT;
MP2_WELLMAP_DEF M3104_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M3105_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M3106_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3107_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3500_Aspirate_Matrix_for_Calibrant;
MP2_WELLMAP_DEF M3501_Dispense_Matrix_for_Calibrant;
MP2_WELLMAP_DEF M3502_AqueousFlush_before_OrganicWash_LVT_1;
MP2_WELLMAP_DEF M3503_AqueousWash_before_OrganicWash_LVT_1;
MP2_WELLMAP_DEF M3504_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M3505_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M3506_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3507_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3800_Aspirate_Matrix_for_Calibrant_optional;
MP2_WELLMAP_DEF M3801_Dispense_Matrix_for_Calibrant_optional;
MP2_WELLMAP_DEF M3802_AqueousFlush_before_OrganicWash_LVT_2;
MP2_WELLMAP_DEF M3803_AqueousWash_before_OrganicWash_LVT_2;
MP2_WELLMAP_DEF M3804_Aspirate_OrganicWash_LVT;
MP2_WELLMAP_DEF M3805_Dispense_OrganicWash_LVT;
MP2_WELLMAP_DEF M3806_AqueousFlush_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M3807_AqueousWash_after_OrganicWash_LVT;
MP2_WELLMAP_DEF M060000_Flush_1;
MP2_WELLMAP_DEF M060001_Wash_1;
MP2_WELLMAP_DEF M060101_Flush_1;
MP2_WELLMAP_DEF M060102_Wash_1;
MP2_WELLMAP_DEF M06010000_Aspirate;
MP2_WELLMAP_DEF M06010001_Dsp1_Dispense;
MP2_WELLMAP_DEF M060201_Flush_1;
```

```
MP2_WELLMAP_DEF M060202_Wash_1;
MP2_WELLMAP_DEF M06020000_Aspirate;
MP2_WELLMAP_DEF M06020001_Dspl_Dispense;
MP2_WELLMAP_DEF M050301_Flush_1;
MP2_WELLMAP_DEF M050302_Wash_1;
MP2_WELLMAP_DEF M05030000_Aspirate;
MP2_WELLMAP_DEF M05030001_Dspl_Dispense;
MP2_WELLMAP_DEF M050401_Flush_1;
MP2_WELLMAP_DEF M050402_Wash_1;
MP2_WELLMAP_DEF M05040000_Aspirate;
MP2_WELLMAP_DEF M05040001_Dspl_Dispense;
MP2_WELLMAP_DEF M1D00_Aspirate_Buffer_2;
MP2_WELLMAP_DEF M1D01_Dispense_Buffer_2;
MP2_WELLMAP_DEF M1D02_AqueousFlush;
MP2_WELLMAP_DEF M1D03_AqueousWash;
MP2_WELLMAP_DEF M1E00_Aspirate_Buffer_2;
MP2_WELLMAP_DEF M1E01_Dispense_Buffer_2;
MP2_WELLMAP_DEF M1E02_AqueousFlush;
MP2_WELLMAP_DEF M1E03_AqueousWash;
MP2_WELLMAP_DEF M2300_Aspirate_Buffer_2;
MP2_WELLMAP_DEF M2301_Dispense_Buffer_2;
MP2_WELLMAP_DEF M2302_AqueousFlush;
MP2_WELLMAP_DEF M2303_AqueousWash;
MP2_WELLMAP_DEF M2400_Aspirate_Buffer_2;
MP2_WELLMAP_DEF M2401_Dispense_Buffer_2;
MP2_WELLMAP_DEF M2402_AqueousFlush;
MP2_WELLMAP_DEF M2403_AqueousWash;
MP2_WELLMAP_DEF M2900_Aspirate_Buffer_2;
MP2_WELLMAP_DEF M2901_Dispense_Buffer_2;
MP2_WELLMAP_DEF M2902_AqueousFlush;
MP2_WELLMAP_DEF M2903_AqueousWash;
MP2_WELLMAP_DEF M2A00_Aspirate_Buffer_2;
MP2_WELLMAP_DEF M2A01_Dispense_Buffer_2;
MP2_WELLMAP_DEF M2A02_AqueousFlush;
MP2_WELLMAP_DEF M2A03_AqueousWash;

int IniAllWellMapDefs()
{
    MSL_IniWellMapDef( &M0C00_Aspirate_Samples, 0, 0, &RtFile1, 2, 3, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M0C01_Dispense_Samples, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M0C04_Aspirate_OrganicWash_LVT, 0, 0, &RtFile6, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M0C05_Dispense_OrganicWash_LVT, 0, 0, &RtFile7, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M0C06_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile8, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M0C07_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile9, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M1B00_Aspirate_Enzyme_1, 0, 0, &RtFile5, 1, 2, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M1B01_Dispense_Enzyme_1, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M1B02_AqueousFlush, 0, 0, &RtFile10, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M1B03_AqueousWash, 0, 0, &RtFile11, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M0200_Pre_Test_Flush_1, 0, 0, &RtFile12, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M0201_Pre_Test_Wash_1, 0, 0, &RtFile13, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2D04_Aspirate_OrganicWash_LVT, 0, 0, &RtFile14, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2D05_Dispense_OrganicWash_LVT, 0, 0, &RtFile15, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2D00_Aspirate_DHB_Matrix_for_Samples, 0, 0, &RtFile5, 5, 6, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2D01_Dispense_DHB_Matrix_for_Samples, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2D06_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile16, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2D07_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile17, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M0E00_Aspirate_Calibrand, 0, 0, &RtFile2, 2, 3, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M0E01_Dispense_Calibrand, 0, 0, &RtFile2, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
}
```

```
MSL_IniWellMapDef( &M0E06_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile18, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0E07_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile19, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3400_Aspirate_Matrix_for_Calibrand, 0, 0, &RtFile5, 5, 6, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3401_Dispense_Matrix_for_Calibrand, 0, 0, &RtFile2, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3404_Aspirate_OrganicWash_LVT, 0, 0, &RtFile20, 1, 2, 4, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3405_Dispense_OrganicWash_LVT, 0, 0, &RtFile21, 1, 2, 4, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3406_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile22, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3407_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile23, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0E04_Aspirate_OrganicWash_LVT, 0, 0, &RtFile24, 1, 2, 4, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0E05_Dispense_OrganicWash_LVT, 0, 0, &RtFile25, 1, 2, 4, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3700_Aspirate_Matrix_for_Calibrand___optional, 0, 0, &RtFile5, 5, 6, 0, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3701_Dispense_Matrix_for_Calibrand___optional, 0, 0, &RtFile2, 26, 27, 0, 1, 1,
0, NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3704_Aspirate_OrganicWash_LVT, 0, 0, &RtFile26, 1, 2, 4, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3705_Dispense_OrganicWash_LVT, 0, 0, &RtFile27, 1, 2, 4, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3706_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile28, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3707_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile29, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3000_Aspirate_Matrix_for_Samples___optional, 0, 0, &RtFile5, 5, 6, 0, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3001_Dispense_Matrix_for_Samples___optional, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3004_Aspirate_OrganicWash_LVT, 0, 0, &RtFile30, 1, 2, 4, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3005_Dispense_OrganicWash_LVT, 0, 0, &RtFile31, 1, 2, 4, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3006_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile32, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3007_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile33, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0C02_AqueousFlush_before_OrganicWash_LVT, 0, 0, &RtFile34, 1, 2, 1, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0C03_AqueousWash_before_OrganicWash_LVT, 0, 0, &RtFile35, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0E02_AqueousFlush_before_OrganicWash_LVT, 0, 0, &RtFile36, 1, 2, 1, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0E03_AqueousWash_before_OrganicWash_LVT, 0, 0, &RtFile37, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2D02_AqueousFlush_before_OrganicWash_LVT, 0, 0, &RtFile38, 1, 2, 1, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2D03_AqueousWash_before_OrganicWash_LVT, 0, 0, &RtFile39, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3003_AqueousWash_before_OrganicWash_LVT, 0, 0, &RtFile40, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3002_AqueousFlush_before_OrganicWash_LVT, 0, 0, &RtFile41, 1, 2, 1, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2100_Aspirate_Enzyme__2, 0, 0, &RtFile5, 1, 3, 0, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2101_Dispense_Enzyme__2, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2102_AqueousFlush, 0, 0, &RtFile42, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2103_AqueousWash, 0, 0, &RtFile43, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2700_Aspirate_Enzyme__3, 0, 0, &RtFile5, 1, 4, 0, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2701_Dispense_Enzyme__3, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2702_AqueousFlush, 0, 0, &RtFile44, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2703_AqueousWash, 0, 0, &RtFile45, 1, 2, 1, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M050101_Flush_1, 0, 0, &RtFile46, 1, 2, 1, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
```



```
MSL_IniWellMapDef( &M050102_Wash_1, 0, 0, &RtFile47, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M05010000_Aspirate, 0, 0, &RtFile4, 9, 10, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M05010001_Dsp1_Dispense, 0, 0, &RtFile5, 9, 10, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0801_Flush_1, 0, 0, &RtFile48, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0802_Wash_1, 0, 0, &RtFile49, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M080000_Aspirate, 0, 0, &RtFile4, 12, 13, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M080001_Dsp1_Dispense, 0, 0, &RtFile5, 13, 14, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M030001_Flush_1, 0, 0, &RtFile50, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M030002_Wash_1, 0, 0, &RtFile51, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M03000000_Aspirate, 0, 0, &RtFile4, 1, 2, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M03000001_Enzyme_1_Dispense, 0, 0, &RtFile5, 1, 2, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M030101_Flush_1, 0, 0, &RtFile52, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M030102_Wash_1, 0, 0, &RtFile53, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M03010000_Aspirate, 0, 0, &RtFile4, 1, 3, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M03010001_Dsp1_Dispense, 0, 0, &RtFile5, 1, 3, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M030201_Flush_1, 0, 0, &RtFile54, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M030202_Wash_1, 0, 0, &RtFile55, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M03020000_Aspirate, 0, 0, &RtFile4, 1, 4, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M03020001_Dsp1_Dispense, 0, 0, &RtFile5, 1, 4, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M050201_Flush_1, 0, 0, &RtFile56, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M050202_Wash_1, 0, 0, &RtFile57, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M05020000_Aspirate, 0, 0, &RtFile4, 9, 11, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M05020001_Dsp1_Dispense, 0, 0, &RtFile5, 11, 12, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M050000_Flush_1, 0, 0, &RtFile58, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M050001_Wash_1, 0, 0, &RtFile59, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
);
MSL_IniWellMapDef( &M1100_Aspirate_Buffer_1, 0, 0, &RtFile5, 9, 10, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1101_Dispense_Buffer_1, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1102_AqueousFlush_before_OrganicWash_LVT, 0, 0, &RtFile60, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1103_AqueousWash_before_OrganicWash_LVT, 0, 0, &RtFile61, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1104_Aspirate_OrganicWash_LVT, 0, 0, &RtFile62, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1105_Dispense_OrganicWash_LVT, 0, 0, &RtFile63, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1106_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile64, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1107_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile65, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1400_Aspirate_Reagent, 0, 0, &RtFile5, 13, 14, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1401_Dispense_Reagent, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1402_AqueousFlush_before_OrganicWash_LVT, 0, 0, &RtFile66, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1403_AqueousWash_before_OrganicWash_LVT, 0, 0, &RtFile67, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1404_Aspirate_OrganicWash_LVT, 0, 0, &RtFile68, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1405_Dispense_OrganicWash_LVT, 0, 0, &RtFile69, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1406_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile70, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
```

```
MSL_IniWellMapDef( &M1407_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile71, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1700_Aspirate_Buffer__2, 0, 0, &RtFile5, 11, 12, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1701_Dispense_Buffer__2, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1702_AqueousFlush_before_OrganicWash_LVT, 0, 0, &RtFile72, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1703_AqueousWash_before_OrganicWash_LVT, 0, 0, &RtFile73, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1704_Aspirate_OrganicWash_LVT, 0, 0, &RtFile74, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1705_Dispense_OrganicWash_LVT, 0, 0, &RtFile75, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1706_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile76, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1707_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile77, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0A00_Flush_1, 0, 0, &RtFile78, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M0A01_Wash_1, 0, 0, &RtFile79, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3402_AqueousFlush_before_OrganicWash_LVT_1, 0, 0, &RtFile80, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3403_AqueousWash_before_OrganicWash_LVT_1, 0, 0, &RtFile81, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3703_AqueousWash_before_OrganicWash_LVT_2, 0, 0, &RtFile82, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3702_AqueousFlush_before_OrganicWash_LVT_2, 0, 0, &RtFile83, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M070002_Flush_1, 0, 0, &RtFile84, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M070003_Wash_1, 0, 0, &RtFile85, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M070000_Aspirate, 0, 0, &RtFile86, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M070001_Dispense, 0, 0, &RtFile87, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M090002_Flush_1, 0, 0, &RtFile88, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M090003_Wash_1, 0, 0, &RtFile89, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M090000_Aspirate, 0, 0, &RtFile90, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M090001_Dispense, 0, 0, &RtFile91, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M2E00_Aspirate_alpha_Matrix_for_Samples, 0, 0, &RtFile5, 7, 8, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2E01_Dispense_alpha_Matrix_for_Samples, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2E02_AqueousFlush_before_OrganicWash_LVT, 0, 0, &RtFile92, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2E03_AqueousWash_before_OrganicWash_LVT, 0, 0, &RtFile93, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2E04_Aspirate_OrganicWash_LVT, 0, 0, &RtFile94, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2E05_Dispense_OrganicWash_LVT, 0, 0, &RtFile95, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2E06_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile96, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M2E07_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile97, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3100_Aspirate_alpha_Matrix_for_Samples, 0, 0, &RtFile5, 7, 8, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3101_Dispense_alpha_Matrix_for_Samples, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3102_AqueousFlush_before_OrganicWash_LVT, 0, 0, &RtFile98, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3103_AqueousWash_before_OrganicWash_LVT, 0, 0, &RtFile99, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3104_Aspirate_OrganicWash_LVT, 0, 0, &RtFile100, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3105_Dispense_OrganicWash_LVT, 0, 0, &RtFile101, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3106_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile102, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3107_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile103, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3500_Aspirate_Matrix_for_Calibrant, 0, 0, &RtFile5, 7, 8, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
```

```
MSL_IniWellMapDef( &M3501_Dispense_Matrix_for_Calibrant, 0, 0, &RtFile2, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3502_AqueousFlush_before_OrganicWash_LVT_1, 0, 0, &RtFile104, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3503_AqueousWash_before_OrganicWash_LVT_1, 0, 0, &RtFile105, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3504_Aspirate_OrganicWash_LVT, 0, 0, &RtFile106, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3505_Dispense_OrganicWash_LVT, 0, 0, &RtFile107, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3506_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile108, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3507_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile109, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3800_Aspirate_Matrix_for_Calibrant___optional, 0, 0, &RtFile5, 7, 8, 0, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3801_Dispense_Matrix_for_Calibrant___optional, 0, 0, &RtFile2, 26, 27, 0, 1, 1,
0, NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3802_AqueousFlush_before_OrganicWash_LVT_2, 0, 0, &RtFile110, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3803_AqueousWash_before_OrganicWash_LVT_2, 0, 0, &RtFile111, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3804_Aspirate_OrganicWash_LVT, 0, 0, &RtFile112, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3805_Dispense_OrganicWash_LVT, 0, 0, &RtFile113, 1, 2, 4, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3806_AqueousFlush_after_OrganicWash_LVT, 0, 0, &RtFile114, 1, 2, 1, 1, 1, 1,
NULL, sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M3807_AqueousWash_after_OrganicWash_LVT, 0, 0, &RtFile115, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M060000_Flush_1, 0, 0, &RtFile116, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M060001_Wash_1, 0, 0, &RtFile117, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M060101_Flush_1, 0, 0, &RtFile118, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M060102_Wash_1, 0, 0, &RtFile119, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M06010000_Aspirate, 0, 0, &RtFile4, 5, 6, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M06010001_Dspl_Dispense, 0, 0, &RtFile5, 5, 6, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M060201_Flush_1, 0, 0, &RtFile120, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M060202_Wash_1, 0, 0, &RtFile121, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M06020000_Aspirate, 0, 0, &RtFile4, 7, 8, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M06020001_Dspl_Dispense, 0, 0, &RtFile5, 7, 8, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M050301_Flush_1, 0, 0, &RtFile122, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M050302_Wash_1, 0, 0, &RtFile123, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M05030000_Aspirate, 0, 0, &RtFile4, 9, 14, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M05030001_Dspl_Dispense, 0, 0, &RtFile5, 15, 16, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M050401_Flush_1, 0, 0, &RtFile124, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M050402_Wash_1, 0, 0, &RtFile125, 1, 2, 1, 1, 1, 1, NULL, sizeof(MP2_WELLMAP_DEF)
);
MSL_IniWellMapDef( &M05040000_Aspirate, 0, 0, &RtFile4, 9, 15, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M05040001_Dspl_Dispense, 0, 0, &RtFile5, 17, 18, 0, 1, 1, 2, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1D00_Aspirate_Buffer_2, 0, 0, &RtFile5, 15, 16, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1D01_Dispense_Buffer_2, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1D02_AqueousFlush, 0, 0, &RtFile126, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1D03_AqueousWash, 0, 0, &RtFile127, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1E00_Aspirate_Buffer_2, 0, 0, &RtFile5, 17, 18, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
MSL_IniWellMapDef( &M1E01_Dispense_Buffer_2, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
```

```

    MSL_IniWellMapDef( &M1E02_AqueousFlush, 0, 0, &RtFile128, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M1E03_AqueousWash, 0, 0, &RtFile129, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2300_Aspirate_Buffer__2, 0, 0, &RtFile5, 15, 16, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2301_Dispense_Buffer__2, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2302_AqueousFlush, 0, 0, &RtFile130, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2303_AqueousWash, 0, 0, &RtFile131, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2400_Aspirate_Buffer__2, 0, 0, &RtFile5, 17, 18, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2401_Dispense_Buffer__2, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2402_AqueousFlush, 0, 0, &RtFile132, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2403_AqueousWash, 0, 0, &RtFile133, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2900_Aspirate_Buffer__2, 0, 0, &RtFile5, 15, 16, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2901_Dispense_Buffer__2, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2902_AqueousFlush, 0, 0, &RtFile134, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2903_AqueousWash, 0, 0, &RtFile135, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2A00_Aspirate_Buffer__2, 0, 0, &RtFile5, 17, 18, 0, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2A01_Dispense_Buffer__2, 0, 0, &RtFile1, 26, 27, 0, 1, 1, 0, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2A02_AqueousFlush, 0, 0, &RtFile136, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );
    MSL_IniWellMapDef( &M2A03_AqueousWash, 0, 0, &RtFile137, 1, 2, 1, 1, 1, 1, NULL,
sizeof(MP2_WELLMAP_DEF) );

    MSL_ReconcileWellMapDefs();

    return 0;
}

/*****
*
*   Procedure Definitions
*
*****/

MP2_PROC_CONTEXT_DEF C00_Verify_Deck_Layout;
MP2_PROC_CONTEXT_DEF C01_Danger_of_Test_Failure_and_LVT_Damage__;
MP2_PROC_CONTEXT_DEF C02_Pre_Test_Flush_Wash;
MP2_PROC_CONTEXT_DEF C03_Enzyme_Panel;
MP2_PROC_CONTEXT_DEF C0300_Enzyme__1_Panel;
MP2_PROC_CONTEXT_DEF C0301_Enzyme__2_Panel;
MP2_PROC_CONTEXT_DEF C0302_Enzyme__3_Panel;
MP2_PROC_CONTEXT_DEF C04_Enzyme_Pre_Incubation;
MP2_PROC_CONTEXT_DEF C05_Buffer_Panel;
MP2_PROC_CONTEXT_DEF C0500_Flush_Wash;
MP2_PROC_CONTEXT_DEF C0501_Buffer__1_Panel;
MP2_PROC_CONTEXT_DEF C0502_Buffer__2_Panel;
MP2_PROC_CONTEXT_DEF C0503_Buffer__3_Panel;
MP2_PROC_CONTEXT_DEF C0504_Buffer__4_Panel;
MP2_PROC_CONTEXT_DEF C06_Matrices_Panel;
MP2_PROC_CONTEXT_DEF C0600_Flush_Wash_1;
MP2_PROC_CONTEXT_DEF C0601_DHB_Matrix_Panel;
MP2_PROC_CONTEXT_DEF C0602_alpha_Matrix_Panel;
MP2_PROC_CONTEXT_DEF C07_Organic_Wash_Matrices_and_Buffers;
MP2_PROC_CONTEXT_DEF C08_Reagent_Panel;
MP2_PROC_CONTEXT_DEF C09_Organic_Wash_Reagent;
MP2_PROC_CONTEXT_DEF C0A_Aqueous_Flush_Wash;
MP2_PROC_CONTEXT_DEF C0B_Sample_Spotting_Time_Marker;
MP2_PROC_CONTEXT_DEF C0C_Samples_to_MALDI26;
MP2_PROC_CONTEXT_DEF C0D_Calibrant_Time_Marker;
MP2_PROC_CONTEXT_DEF C0E_Calibrant_to_MALDI26;
MP2_PROC_CONTEXT_DEF C0F_Sample_Drying_Timer;
MP2_PROC_CONTEXT_DEF C10_Buffer__1_Spotting_Time_Marker;

```

```

MP2_PROC_CONTEXT_DEF C11_Buffer_1_to_MALDI26;
MP2_PROC_CONTEXT_DEF C12_Buffer_1_Timer;
MP2_PROC_CONTEXT_DEF C13_Reagent_Spotting_Time_Marker;
MP2_PROC_CONTEXT_DEF C14_Reagent_to_MALDI26;
MP2_PROC_CONTEXT_DEF C15_Reaction_Timer;
MP2_PROC_CONTEXT_DEF C16_Buffer_2_Spotting_Time_Marker;
MP2_PROC_CONTEXT_DEF C17_Buffer_2_to_MALDI26;
MP2_PROC_CONTEXT_DEF C18_Buffer_2_Timer;
MP2_PROC_CONTEXT_DEF C19_Enzyme_Pre_Incubation_Time;
MP2_PROC_CONTEXT_DEF C1A_Enzyme_1_RTM;
MP2_PROC_CONTEXT_DEF C1B_Enzyme_1_to_MALDI26;
MP2_PROC_CONTEXT_DEF C1C_Enzyme_1_IT;
MP2_PROC_CONTEXT_DEF C1D_Buffer_3_E1_to_MALDI26;
MP2_PROC_CONTEXT_DEF C1E_Buffer_4_E1_to_MALDI26;
MP2_PROC_CONTEXT_DEF C1F_Enzyme_1_RT;
MP2_PROC_CONTEXT_DEF C20_Enzyme_2_RTM;
MP2_PROC_CONTEXT_DEF C21_Enzyme_2_to_MALDI26;
MP2_PROC_CONTEXT_DEF C22_Enzyme_2_IT;
MP2_PROC_CONTEXT_DEF C23_Buffer_3_E2_to_MALDI26;
MP2_PROC_CONTEXT_DEF C24_Buffer_4_E2_to_MALDI26;
MP2_PROC_CONTEXT_DEF C25_Enzyme_2_RT;
MP2_PROC_CONTEXT_DEF C26_Enzyme_3_RTM;
MP2_PROC_CONTEXT_DEF C27_Enzyme_3_to_MALDI26;
MP2_PROC_CONTEXT_DEF C28_Enzyme_3_IT;
MP2_PROC_CONTEXT_DEF C29_Buffer_3_E3_to_MALDI26;
MP2_PROC_CONTEXT_DEF C2A_Buffer_4_E3_to_MALDI26;
MP2_PROC_CONTEXT_DEF C2B_Enzyme_3_RT;
MP2_PROC_CONTEXT_DEF C2C_Sample_Matrix_Time_Marker;
MP2_PROC_CONTEXT_DEF C2D_DHB_Matrix_to_Samples_on_MALDI26;
MP2_PROC_CONTEXT_DEF C2E_alpha_Matrix_to_Samples_on_MALDI26;
MP2_PROC_CONTEXT_DEF C2F_Sample_Matrix_Timer;
MP2_PROC_CONTEXT_DEF C30_DHB_Matrix_to_Samples_on_MALDI26_optional;
MP2_PROC_CONTEXT_DEF C31_alpha_Matrix_to_Samples_on_MALDI26_optional;
MP2_PROC_CONTEXT_DEF C32_Calibrant_Timer;
MP2_PROC_CONTEXT_DEF C33_Calibrant_Matrix_Time_Marker;
MP2_PROC_CONTEXT_DEF C34_DHB_Matrix_to_Calibrant_on_MALDI26;
MP2_PROC_CONTEXT_DEF C35_alpha_Matrix_to_Calibrant_on_MALDI26;
MP2_PROC_CONTEXT_DEF C36_Calibrant_Matrix_Timer;
MP2_PROC_CONTEXT_DEF C37_DHB_Matrix_to_Calibrant_on_MALDI26_optional;
MP2_PROC_CONTEXT_DEF C38_alpha_Matrix_to_Calibrant_on_MALDI26_optional;
MP2_PROC_CONTEXT_DEF C39_FTP_Transfer_to_MALDI_Reflex_III;
MP2_PROC_CONTEXT_DEF C3A_User_Information;

int IniAllProcDefs()
{
    MSL_IniProcDef( &C00_Verify_Deck_Layout, "Verify Deck-Layout", 1, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C01_Danger_of_Test_Failiure_and_LVT_Damage__, "Danger of Test Failiure and LVT-Damage
!", 3, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C02_Pre_Test_Flush_Wash, "Pre-Test Flush/Wash", 5, 2, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C02_Pre_Test_Flush_Wash, &M0200_Pre_Test_Flush_1, 'W', 0 );
    MSL_AddProcWellMapRef( &C02_Pre_Test_Flush_Wash, &M0201_Pre_Test_Wash_1, 'W', 0 );
    MSL_IniProcDef( &C03_Enzyme_Panel, "Enzyme Panel", 13, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C0300_Enzyme_1_Panel, "Enzyme #1 Panel", 14, 4, &C03_Enzyme_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0300_Enzyme_1_Panel, &M03000000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C0300_Enzyme_1_Panel, &M03000001_Enzyme_1_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C0300_Enzyme_1_Panel, &M030001_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C0300_Enzyme_1_Panel, &M030002_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C0301_Enzyme_2_Panel, "Enzyme #2 Panel", 26, 4, &C03_Enzyme_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0301_Enzyme_2_Panel, &M03010000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C0301_Enzyme_2_Panel, &M03010001_Dspl1_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C0301_Enzyme_2_Panel, &M030101_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C0301_Enzyme_2_Panel, &M030102_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C0302_Enzyme_3_Panel, "Enzyme #3 Panel", 38, 4, &C03_Enzyme_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0302_Enzyme_3_Panel, &M03020000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C0302_Enzyme_3_Panel, &M03020001_Dspl1_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C0302_Enzyme_3_Panel, &M030201_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C0302_Enzyme_3_Panel, &M030202_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C04_Enzyme_Pre_Incubation, "Enzyme Pre-Incubation", 51, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C05_Buffer_Panel, "Buffer Panel", 53, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C0500_Flush_Wash, "Flush/Wash", 54, 2, &C05_Buffer_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0500_Flush_Wash, &M050000_Flush_1, 'W', 0 );

```

```

    MSL_AddProcWellMapRef( &C0500_Flush_Wash, &M050001_Wash_1, 'W', 0 );
    MSL_IniProcDef( &C0501_Buffer_1_Panel, "Buffer #1 Panel", 62, 4, &C05_Buffer_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0501_Buffer_1_Panel, &M05010000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C0501_Buffer_1_Panel, &M05010001_Dspl_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C0501_Buffer_1_Panel, &M050101_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C0501_Buffer_1_Panel, &M050102_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C0502_Buffer_2_Panel, "Buffer #2 Panel", 74, 4, &C05_Buffer_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0502_Buffer_2_Panel, &M05020000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C0502_Buffer_2_Panel, &M05020001_Dspl_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C0502_Buffer_2_Panel, &M050201_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C0502_Buffer_2_Panel, &M050202_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C0503_Buffer_3_Panel, "Buffer #3 Panel", 86, 4, &C05_Buffer_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0503_Buffer_3_Panel, &M05030000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C0503_Buffer_3_Panel, &M05030001_Dspl_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C0503_Buffer_3_Panel, &M050301_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C0503_Buffer_3_Panel, &M050302_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C0504_Buffer_4_Panel, "Buffer #4 Panel", 98, 4, &C05_Buffer_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0504_Buffer_4_Panel, &M05040000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C0504_Buffer_4_Panel, &M05040001_Dspl_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C0504_Buffer_4_Panel, &M050401_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C0504_Buffer_4_Panel, &M050402_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C06_Matrices_Panel, "Matrices Panel", 111, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C0600_Flush_Wash_1, "Flush/Wash_1", 112, 2, &C06_Matrices_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0600_Flush_Wash_1, &M060000_Flush_1, 'W', 0 );
    MSL_AddProcWellMapRef( &C0600_Flush_Wash_1, &M060001_Wash_1, 'W', 0 );
    MSL_IniProcDef( &C0601_DHB_Matrix_Panel, "DHB-Matrix Panel", 120, 4, &C06_Matrices_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0601_DHB_Matrix_Panel, &M06010000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C0601_DHB_Matrix_Panel, &M06010001_Dspl_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C0601_DHB_Matrix_Panel, &M060101_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C0601_DHB_Matrix_Panel, &M060102_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C0602_alpha_Matrix_Panel, "alpha-Matrix Panel", 132, 4, &C06_Matrices_Panel,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0602_alpha_Matrix_Panel, &M06020000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C0602_alpha_Matrix_Panel, &M06020001_Dspl_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C0602_alpha_Matrix_Panel, &M060201_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C0602_alpha_Matrix_Panel, &M060202_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C07_Organic_Wash_Matrices_and_Buffers, "Organic Wash Matrices and Buffers", 145, 4,
NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C07_Organic_Wash_Matrices_and_Buffers, &M070000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C07_Organic_Wash_Matrices_and_Buffers, &M070001_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C07_Organic_Wash_Matrices_and_Buffers, &M070002_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C07_Organic_Wash_Matrices_and_Buffers, &M070003_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C08_Reagent_Panel, "Reagent Panel", 161, 4, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C08_Reagent_Panel, &M080000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C08_Reagent_Panel, &M080001_Dspl_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C08_Reagent_Panel, &M0801_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C08_Reagent_Panel, &M0802_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C09_Organic_Wash_Reagent, "Organic Wash Reagent", 173, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C09_Organic_Wash_Reagent, &M090000_Aspirate, 'A', 1 );
    MSL_AddProcWellMapRef( &C09_Organic_Wash_Reagent, &M090001_Dispense, 'D', 0 );
    MSL_AddProcWellMapRef( &C09_Organic_Wash_Reagent, &M090002_Flush_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C09_Organic_Wash_Reagent, &M090003_Wash_1, 'W', 1 );
    MSL_IniProcDef( &C0A_Aqueous_Flush_Wash, "Aqueous Flush/Wash", 189, 2, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0A_Aqueous_Flush_Wash, &M0A00_Flush_1, 'W', 0 );
    MSL_AddProcWellMapRef( &C0A_Aqueous_Flush_Wash, &M0A01_Wash_1, 'W', 0 );
    MSL_IniProcDef( &C0B_Sample_Spotting_Time_Marker, "Sample Spotting Time Marker", 197, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C0C_Samples_to_MALDI26, "Samples to MALDI26", 199, 8, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C0C_Samples_to_MALDI26, &M0C00_Aspirate_Samples, 'A', 2 );
    MSL_AddProcWellMapRef( &C0C_Samples_to_MALDI26, &M0C01_Dispense_Samples, 'D', 1 );
    MSL_AddProcWellMapRef( &C0C_Samples_to_MALDI26, &M0C02_AqueousFlush_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C0C_Samples_to_MALDI26, &M0C03_AqueousWash_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C0C_Samples_to_MALDI26, &M0C04_Aspirate_OrganicWash_LVT, 'A', 1 );
    MSL_AddProcWellMapRef( &C0C_Samples_to_MALDI26, &M0C05_Dispense_OrganicWash_LVT, 'D', 2 );
    MSL_AddProcWellMapRef( &C0C_Samples_to_MALDI26, &M0C06_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C0C_Samples_to_MALDI26, &M0C07_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
    MSL_IniProcDef( &C0D_Calibrant_Time_Marker, "Calibrant Time Marker", 221, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );

```

```
MSL_IniProcDef( &C0E_Calibrant_to_MALDI26, "Calibrant to MALDI26", 223, 8, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_AddProcWellMapRef( &C0E_Calibrant_to_MALDI26, &M0E00_Aspirate_Calibrant, 'A', 1 );
MSL_AddProcWellMapRef( &C0E_Calibrant_to_MALDI26, &M0E01_Dispense_Calibrant, 'D', 2 );
MSL_AddProcWellMapRef( &C0E_Calibrant_to_MALDI26, &M0E02_AqueousFlush_before_OrganicWash_LVT, 'W', 1
);
MSL_AddProcWellMapRef( &C0E_Calibrant_to_MALDI26, &M0E03_AqueousWash_before_OrganicWash_LVT, 'W', 1 );
MSL_AddProcWellMapRef( &C0E_Calibrant_to_MALDI26, &M0E04_Aspirate_OrganicWash_LVT, 'A', 1 );
MSL_AddProcWellMapRef( &C0E_Calibrant_to_MALDI26, &M0E05_Dispense_OrganicWash_LVT, 'D', 0 );
MSL_AddProcWellMapRef( &C0E_Calibrant_to_MALDI26, &M0E06_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
MSL_AddProcWellMapRef( &C0E_Calibrant_to_MALDI26, &M0E07_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
MSL_IniProcDef( &C0F_Sample_Drying_Timer, "Sample Drying Timer", 245, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C10_Buffer_1_Spotting_Time_Marker, "Buffer #1 Spotting Time Marker", 247, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C11_Buffer_1_to_MALDI26, "Buffer #1 to MALDI26", 249, 8, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_AddProcWellMapRef( &C11_Buffer_1_to_MALDI26, &M1100_Aspirate_Buffer_1, 'A', 2 );
MSL_AddProcWellMapRef( &C11_Buffer_1_to_MALDI26, &M1101_Dispense_Buffer_1, 'D', 1 );
MSL_AddProcWellMapRef( &C11_Buffer_1_to_MALDI26, &M1102_AqueousFlush_before_OrganicWash_LVT, 'W', 1
);
MSL_AddProcWellMapRef( &C11_Buffer_1_to_MALDI26, &M1103_AqueousWash_before_OrganicWash_LVT, 'W', 1 );
MSL_AddProcWellMapRef( &C11_Buffer_1_to_MALDI26, &M1104_Aspirate_OrganicWash_LVT, 'A', 1 );
MSL_AddProcWellMapRef( &C11_Buffer_1_to_MALDI26, &M1105_Dispense_OrganicWash_LVT, 'D', 2 );
MSL_AddProcWellMapRef( &C11_Buffer_1_to_MALDI26, &M1106_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
MSL_AddProcWellMapRef( &C11_Buffer_1_to_MALDI26, &M1107_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
MSL_IniProcDef( &C12_Buffer_1_Timer, "Buffer #1 Timer", 271, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C13_Reagent_Spotting_Time_Marker, "Reagent Spotting Time Marker", 273, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C14_Reagent_to_MALDI26, "Reagent to MALDI26", 275, 8, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_AddProcWellMapRef( &C14_Reagent_to_MALDI26, &M1400_Aspirate_Reagent, 'A', 2 );
MSL_AddProcWellMapRef( &C14_Reagent_to_MALDI26, &M1401_Dispense_Reagent, 'D', 1 );
MSL_AddProcWellMapRef( &C14_Reagent_to_MALDI26, &M1402_AqueousFlush_before_OrganicWash_LVT, 'W', 1 );
MSL_AddProcWellMapRef( &C14_Reagent_to_MALDI26, &M1403_AqueousWash_before_OrganicWash_LVT, 'W', 1 );
MSL_AddProcWellMapRef( &C14_Reagent_to_MALDI26, &M1404_Aspirate_OrganicWash_LVT, 'A', 1 );
MSL_AddProcWellMapRef( &C14_Reagent_to_MALDI26, &M1405_Dispense_OrganicWash_LVT, 'D', 2 );
MSL_AddProcWellMapRef( &C14_Reagent_to_MALDI26, &M1406_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
MSL_AddProcWellMapRef( &C14_Reagent_to_MALDI26, &M1407_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
MSL_IniProcDef( &C15_Reaction_Timer, "Reaction Timer", 297, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C16_Buffer_2_Spotting_Time_Marker, "Buffer #2 Spotting Time Marker", 299, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C17_Buffer_2_to_MALDI26, "Buffer #2 to MALDI26", 301, 8, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_AddProcWellMapRef( &C17_Buffer_2_to_MALDI26, &M1700_Aspirate_Buffer_2, 'A', 2 );
MSL_AddProcWellMapRef( &C17_Buffer_2_to_MALDI26, &M1701_Dispense_Buffer_2, 'D', 1 );
MSL_AddProcWellMapRef( &C17_Buffer_2_to_MALDI26, &M1702_AqueousFlush_before_OrganicWash_LVT, 'W', 1
);
MSL_AddProcWellMapRef( &C17_Buffer_2_to_MALDI26, &M1703_AqueousWash_before_OrganicWash_LVT, 'W', 1 );
MSL_AddProcWellMapRef( &C17_Buffer_2_to_MALDI26, &M1704_Aspirate_OrganicWash_LVT, 'A', 1 );
MSL_AddProcWellMapRef( &C17_Buffer_2_to_MALDI26, &M1705_Dispense_OrganicWash_LVT, 'D', 2 );
MSL_AddProcWellMapRef( &C17_Buffer_2_to_MALDI26, &M1706_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
MSL_AddProcWellMapRef( &C17_Buffer_2_to_MALDI26, &M1707_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
MSL_IniProcDef( &C18_Buffer_2_Timer, "Buffer #2 Timer", 323, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C19_Enzyme_Pre_Incubation_Time, "Enzyme Pre-Incubation Time", 325, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C1A_Enzyme_1_RTM, "Enzyme 1 RTM", 327, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C1B_Enzyme_1_to_MALDI26, "Enzyme #1 to MALDI26", 329, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_AddProcWellMapRef( &C1B_Enzyme_1_to_MALDI26, &M1B00_Aspirate_Enzyme_1, 'A', 0 );
MSL_AddProcWellMapRef( &C1B_Enzyme_1_to_MALDI26, &M1B01_Dispense_Enzyme_1, 'D', 3 );
MSL_AddProcWellMapRef( &C1B_Enzyme_1_to_MALDI26, &M1B02_AqueousFlush, 'W', 1 );
MSL_AddProcWellMapRef( &C1B_Enzyme_1_to_MALDI26, &M1B03_AqueousWash, 'W', 1 );
MSL_IniProcDef( &C1C_Enzyme_1_IT, "Enzyme 1 IT", 339, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C1D_Buffer_3_E1_to_MALDI26, "Buffer #3 (E1) to MALDI26", 341, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_AddProcWellMapRef( &C1D_Buffer_3_E1_to_MALDI26, &M1D00_Aspirate_Buffer_2, 'A', 2 );
MSL_AddProcWellMapRef( &C1D_Buffer_3_E1_to_MALDI26, &M1D01_Dispense_Buffer_2, 'D', 1 );
MSL_AddProcWellMapRef( &C1D_Buffer_3_E1_to_MALDI26, &M1D02_AqueousFlush, 'W', 1 );
MSL_AddProcWellMapRef( &C1D_Buffer_3_E1_to_MALDI26, &M1D03_AqueousWash, 'W', 1 );
MSL_IniProcDef( &C1E_Buffer_4_E1_to_MALDI26, "Buffer #4 (E1) to MALDI26", 351, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_AddProcWellMapRef( &C1E_Buffer_4_E1_to_MALDI26, &M1E00_Aspirate_Buffer_2, 'A', 2 );
MSL_AddProcWellMapRef( &C1E_Buffer_4_E1_to_MALDI26, &M1E01_Dispense_Buffer_2, 'D', 1 );
MSL_AddProcWellMapRef( &C1E_Buffer_4_E1_to_MALDI26, &M1E02_AqueousFlush, 'W', 1 );
MSL_AddProcWellMapRef( &C1E_Buffer_4_E1_to_MALDI26, &M1E03_AqueousWash, 'W', 1 );
MSL_IniProcDef( &C1F_Enzyme_1_RT, "Enzyme 1 RT", 361, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
MSL_IniProcDef( &C20_Enzyme_2_RTM, "Enzyme 2 RTM", 363, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
```

```

    MSL_IniProcDef( &C21_Enzyme_2_to_MALDI26, "Enzyme #2 to MALDI26", 365, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C21_Enzyme_2_to_MALDI26, &M2100_Aspirate_Enzyme_2, 'A', 0 );
    MSL_AddProcWellMapRef( &C21_Enzyme_2_to_MALDI26, &M2101_Dispense_Enzyme_2, 'D', 3 );
    MSL_AddProcWellMapRef( &C21_Enzyme_2_to_MALDI26, &M2102_AqueousFlush, 'W', 1 );
    MSL_AddProcWellMapRef( &C21_Enzyme_2_to_MALDI26, &M2103_AqueousWash, 'W', 1 );
    MSL_IniProcDef( &C22_Enzyme_2_IT, "Enzyme 2 IT", 375, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C23_Buffer_3_E2_to_MALDI26, "Buffer #3 (E2) to MALDI26", 377, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C23_Buffer_3_E2_to_MALDI26, &M2300_Aspirate_Buffer_2, 'A', 2 );
    MSL_AddProcWellMapRef( &C23_Buffer_3_E2_to_MALDI26, &M2301_Dispense_Buffer_2, 'D', 1 );
    MSL_AddProcWellMapRef( &C23_Buffer_3_E2_to_MALDI26, &M2302_AqueousFlush, 'W', 1 );
    MSL_AddProcWellMapRef( &C23_Buffer_3_E2_to_MALDI26, &M2303_AqueousWash, 'W', 1 );
    MSL_IniProcDef( &C24_Buffer_4_E2_to_MALDI26, "Buffer #4 (E2) to MALDI26", 387, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C24_Buffer_4_E2_to_MALDI26, &M2400_Aspirate_Buffer_2, 'A', 2 );
    MSL_AddProcWellMapRef( &C24_Buffer_4_E2_to_MALDI26, &M2401_Dispense_Buffer_2, 'D', 1 );
    MSL_AddProcWellMapRef( &C24_Buffer_4_E2_to_MALDI26, &M2402_AqueousFlush, 'W', 1 );
    MSL_AddProcWellMapRef( &C24_Buffer_4_E2_to_MALDI26, &M2403_AqueousWash, 'W', 1 );
    MSL_IniProcDef( &C25_Enzyme_2_RT, "Enzyme 2 RT", 397, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C26_Enzyme_3_RTM, "Enzyme 3 RTM", 399, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C27_Enzyme_3_to_MALDI26, "Enzyme #3 to MALDI26", 401, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C27_Enzyme_3_to_MALDI26, &M2700_Aspirate_Enzyme_3, 'A', 0 );
    MSL_AddProcWellMapRef( &C27_Enzyme_3_to_MALDI26, &M2701_Dispense_Enzyme_3, 'D', 3 );
    MSL_AddProcWellMapRef( &C27_Enzyme_3_to_MALDI26, &M2702_AqueousFlush, 'W', 1 );
    MSL_AddProcWellMapRef( &C27_Enzyme_3_to_MALDI26, &M2703_AqueousWash, 'W', 1 );
    MSL_IniProcDef( &C28_Enzyme_3_IT, "Enzyme 3 IT", 411, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C29_Buffer_3_E3_to_MALDI26, "Buffer #3 (E3) to MALDI26", 413, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C29_Buffer_3_E3_to_MALDI26, &M2900_Aspirate_Buffer_2, 'A', 2 );
    MSL_AddProcWellMapRef( &C29_Buffer_3_E3_to_MALDI26, &M2901_Dispense_Buffer_2, 'D', 1 );
    MSL_AddProcWellMapRef( &C29_Buffer_3_E3_to_MALDI26, &M2902_AqueousFlush, 'W', 1 );
    MSL_AddProcWellMapRef( &C29_Buffer_3_E3_to_MALDI26, &M2903_AqueousWash, 'W', 1 );
    MSL_IniProcDef( &C2A_Buffer_4_E3_to_MALDI26, "Buffer #4 (E3) to MALDI26", 423, 4, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C2A_Buffer_4_E3_to_MALDI26, &M2A00_Aspirate_Buffer_2, 'A', 2 );
    MSL_AddProcWellMapRef( &C2A_Buffer_4_E3_to_MALDI26, &M2A01_Dispense_Buffer_2, 'D', 1 );
    MSL_AddProcWellMapRef( &C2A_Buffer_4_E3_to_MALDI26, &M2A02_AqueousFlush, 'W', 1 );
    MSL_AddProcWellMapRef( &C2A_Buffer_4_E3_to_MALDI26, &M2A03_AqueousWash, 'W', 1 );
    MSL_IniProcDef( &C2B_Enzyme_3_RT, "Enzyme 3 RT", 433, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C2C_Sample_Matrix_Time_Marker, "Sample Matrix Time Marker", 435, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C2D_DHB_Matrix_to_Samples_on_MALDI26, "DHB Matrix to Samples on MALDI26", 437, 8,
NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C2D_DHB_Matrix_to_Samples_on_MALDI26, &M2D00_Aspirate_DHB_Matrix_for_Samples,
'A', 1 );
    MSL_AddProcWellMapRef( &C2D_DHB_Matrix_to_Samples_on_MALDI26, &M2D01_Dispense_DHB_Matrix_for_Samples,
'D', 3 );
    MSL_AddProcWellMapRef( &C2D_DHB_Matrix_to_Samples_on_MALDI26,
&M2D02_AqueousFlush_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C2D_DHB_Matrix_to_Samples_on_MALDI26,
&M2D03_AqueousWash_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C2D_DHB_Matrix_to_Samples_on_MALDI26, &M2D04_Aspirate_OrganicWash_LVT, 'A', 1
);
    MSL_AddProcWellMapRef( &C2D_DHB_Matrix_to_Samples_on_MALDI26, &M2D05_Dispense_OrganicWash_LVT, 'D', 0
);
    MSL_AddProcWellMapRef( &C2D_DHB_Matrix_to_Samples_on_MALDI26,
&M2D06_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C2D_DHB_Matrix_to_Samples_on_MALDI26,
&M2D07_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
    MSL_IniProcDef( &C2E_alpha_Matrix_to_Samples_on_MALDI26, "alpha Matrix to Samples on MALDI26", 459, 8,
NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C2E_alpha_Matrix_to_Samples_on_MALDI26,
&M2E00_Aspirate_alpha_Matrix_for_Samples, 'A', 1 );
    MSL_AddProcWellMapRef( &C2E_alpha_Matrix_to_Samples_on_MALDI26,
&M2E01_Dispense_alpha_Matrix_for_Samples, 'D', 3 );
    MSL_AddProcWellMapRef( &C2E_alpha_Matrix_to_Samples_on_MALDI26,
&M2E02_AqueousFlush_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C2E_alpha_Matrix_to_Samples_on_MALDI26,
&M2E03_AqueousWash_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C2E_alpha_Matrix_to_Samples_on_MALDI26, &M2E04_Aspirate_OrganicWash_LVT, 'A',
1 );
    MSL_AddProcWellMapRef( &C2E_alpha_Matrix_to_Samples_on_MALDI26, &M2E05_Dispense_OrganicWash_LVT, 'D',
0 );
    MSL_AddProcWellMapRef( &C2E_alpha_Matrix_to_Samples_on_MALDI26,
&M2E06_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );

```



```

    MSL_AddProcWellMapRef( &C2E_alpha_Matrix_to_Samples_on_MALDI26,
&M2E07_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
    MSL_IniProcDef( &C2F_Sample_Matrix_Timer, "Sample Matrix Timer", 481, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C30_DHB_Matrix_to_Samples_on_MALDI26___optional, "DHB Matrix to Samples on MALDI26 -
optional", 483, 8, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C30_DHB_Matrix_to_Samples_on_MALDI26___optional,
&M3000_Aspirate_Matrix_for_Samples___optional, 'A', 1 );
    MSL_AddProcWellMapRef( &C30_DHB_Matrix_to_Samples_on_MALDI26___optional,
&M3001_Dispende_Matrix_for_Samples___optional, 'D', 3 );
    MSL_AddProcWellMapRef( &C30_DHB_Matrix_to_Samples_on_MALDI26___optional,
&M3002_AqueousFlush_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C30_DHB_Matrix_to_Samples_on_MALDI26___optional,
&M3003_AqueousWash_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C30_DHB_Matrix_to_Samples_on_MALDI26___optional,
&M3004_Aspirate_OrganicWash_LVT, 'A', 1 );
    MSL_AddProcWellMapRef( &C30_DHB_Matrix_to_Samples_on_MALDI26___optional,
&M3005_Dispende_OrganicWash_LVT, 'D', 0 );
    MSL_AddProcWellMapRef( &C30_DHB_Matrix_to_Samples_on_MALDI26___optional,
&M3006_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C30_DHB_Matrix_to_Samples_on_MALDI26___optional,
&M3007_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
    MSL_IniProcDef( &C31_alpha_Matrix_to_Samples_on_MALDI26___optional, "alpha Matrix to Samples on
MALDI26 - optional", 505, 8, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C31_alpha_Matrix_to_Samples_on_MALDI26___optional,
&M3100_Aspirate_alpha_Matrix_for_Samples, 'A', 1 );
    MSL_AddProcWellMapRef( &C31_alpha_Matrix_to_Samples_on_MALDI26___optional,
&M3101_Dispende_alpha_Matrix_for_Samples, 'D', 3 );
    MSL_AddProcWellMapRef( &C31_alpha_Matrix_to_Samples_on_MALDI26___optional,
&M3102_AqueousFlush_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C31_alpha_Matrix_to_Samples_on_MALDI26___optional,
&M3103_AqueousWash_before_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C31_alpha_Matrix_to_Samples_on_MALDI26___optional,
&M3104_Aspirate_OrganicWash_LVT, 'A', 1 );
    MSL_AddProcWellMapRef( &C31_alpha_Matrix_to_Samples_on_MALDI26___optional,
&M3105_Dispende_OrganicWash_LVT, 'D', 0 );
    MSL_AddProcWellMapRef( &C31_alpha_Matrix_to_Samples_on_MALDI26___optional,
&M3106_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C31_alpha_Matrix_to_Samples_on_MALDI26___optional,
&M3107_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
    MSL_IniProcDef( &C32_Calibrand_Timer, "Calibrand Timer", 527, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C33_Calibrand_Matrix_Time_Marker, "Calibrand Matrix Time Marker", 529, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C34_DHB_Matrix_to_Calibrand_on_MALDI26, "DHB Matrix to Calibrand on MALDI26", 531, 8,
NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C34_DHB_Matrix_to_Calibrand_on_MALDI26, &M3400_Aspirate_Matrix_for_Calibrand,
'A', 2 );
    MSL_AddProcWellMapRef( &C34_DHB_Matrix_to_Calibrand_on_MALDI26, &M3401_Dispende_Matrix_for_Calibrand,
'D', 3 );
    MSL_AddProcWellMapRef( &C34_DHB_Matrix_to_Calibrand_on_MALDI26,
&M3402_AqueousFlush_before_OrganicWash_LVT_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C34_DHB_Matrix_to_Calibrand_on_MALDI26,
&M3403_AqueousWash_before_OrganicWash_LVT_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C34_DHB_Matrix_to_Calibrand_on_MALDI26, &M3404_Aspirate_OrganicWash_LVT, 'A',
1 );
    MSL_AddProcWellMapRef( &C34_DHB_Matrix_to_Calibrand_on_MALDI26, &M3405_Dispende_OrganicWash_LVT, 'D',
0 );
    MSL_AddProcWellMapRef( &C34_DHB_Matrix_to_Calibrand_on_MALDI26,
&M3406_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C34_DHB_Matrix_to_Calibrand_on_MALDI26,
&M3407_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
    MSL_IniProcDef( &C35_alpha_Matrix_to_Calibrand_on_MALDI26, "alpha Matrix to Calibrand on MALDI26",
553, 8, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C35_alpha_Matrix_to_Calibrand_on_MALDI26,
&M3500_Aspirate_Matrix_for_Calibrand, 'A', 2 );
    MSL_AddProcWellMapRef( &C35_alpha_Matrix_to_Calibrand_on_MALDI26,
&M3501_Dispende_Matrix_for_Calibrand, 'D', 3 );
    MSL_AddProcWellMapRef( &C35_alpha_Matrix_to_Calibrand_on_MALDI26,
&M3502_AqueousFlush_before_OrganicWash_LVT_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C35_alpha_Matrix_to_Calibrand_on_MALDI26,
&M3503_AqueousWash_before_OrganicWash_LVT_1, 'W', 1 );
    MSL_AddProcWellMapRef( &C35_alpha_Matrix_to_Calibrand_on_MALDI26, &M3504_Aspirate_OrganicWash_LVT,
'A', 1 );
    MSL_AddProcWellMapRef( &C35_alpha_Matrix_to_Calibrand_on_MALDI26, &M3505_Dispende_OrganicWash_LVT,
'D', 0 );
    MSL_AddProcWellMapRef( &C35_alpha_Matrix_to_Calibrand_on_MALDI26,
&M3506_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );

```

```

    MSL_AddProcWellMapRef( &C35_alpha_Matrix_to_Calibrant_on_MALDI26,
&M3507_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
    MSL_IniProcDef( &C36_Calibrant_Matrix_Timer, "Calibrant Matrix Timer", 575, 0, NULL,
sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C37_DHB_Matrix_to_Calibrant_on_MALDI26__optional, "DHB Matrix to Calibrant on
MALDI26 - optional", 577, 8, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C37_DHB_Matrix_to_Calibrant_on_MALDI26__optional,
&M3700_Aspirate_Matrix_for_Calibrant__optional, 'A', 2 );
    MSL_AddProcWellMapRef( &C37_DHB_Matrix_to_Calibrant_on_MALDI26__optional,
&M3701_Disperse_Matrix_for_Calibrant__optional, 'D', 3 );
    MSL_AddProcWellMapRef( &C37_DHB_Matrix_to_Calibrant_on_MALDI26__optional,
&M3702_AqueousFlush_before_OrganicWash_LVT_2, 'W', 1 );
    MSL_AddProcWellMapRef( &C37_DHB_Matrix_to_Calibrant_on_MALDI26__optional,
&M3703_AqueousWash_before_OrganicWash_LVT_2, 'W', 1 );
    MSL_AddProcWellMapRef( &C37_DHB_Matrix_to_Calibrant_on_MALDI26__optional,
&M3704_Aspirate_OrganicWash_LVT, 'A', 1 );
    MSL_AddProcWellMapRef( &C37_DHB_Matrix_to_Calibrant_on_MALDI26__optional,
&M3705_Disperse_OrganicWash_LVT, 'D', 0 );
    MSL_AddProcWellMapRef( &C37_DHB_Matrix_to_Calibrant_on_MALDI26__optional,
&M3706_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C37_DHB_Matrix_to_Calibrant_on_MALDI26__optional,
&M3707_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
    MSL_IniProcDef( &C38_alpha_Matrix_to_Calibrant_on_MALDI26__optional, "alpha Matrix to Calibrant on
MALDI26 - optional", 599, 8, NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_AddProcWellMapRef( &C38_alpha_Matrix_to_Calibrant_on_MALDI26__optional,
&M3800_Aspirate_Matrix_for_Calibrant__optional, 'A', 2 );
    MSL_AddProcWellMapRef( &C38_alpha_Matrix_to_Calibrant_on_MALDI26__optional,
&M3801_Disperse_Matrix_for_Calibrant__optional, 'D', 3 );
    MSL_AddProcWellMapRef( &C38_alpha_Matrix_to_Calibrant_on_MALDI26__optional,
&M3802_AqueousFlush_before_OrganicWash_LVT_2, 'W', 1 );
    MSL_AddProcWellMapRef( &C38_alpha_Matrix_to_Calibrant_on_MALDI26__optional,
&M3803_AqueousWash_before_OrganicWash_LVT_2, 'W', 1 );
    MSL_AddProcWellMapRef( &C38_alpha_Matrix_to_Calibrant_on_MALDI26__optional,
&M3804_Aspirate_OrganicWash_LVT, 'A', 1 );
    MSL_AddProcWellMapRef( &C38_alpha_Matrix_to_Calibrant_on_MALDI26__optional,
&M3805_Disperse_OrganicWash_LVT, 'D', 0 );
    MSL_AddProcWellMapRef( &C38_alpha_Matrix_to_Calibrant_on_MALDI26__optional,
&M3806_AqueousFlush_after_OrganicWash_LVT, 'W', 1 );
    MSL_AddProcWellMapRef( &C38_alpha_Matrix_to_Calibrant_on_MALDI26__optional,
&M3807_AqueousWash_after_OrganicWash_LVT, 'W', 1 );
    MSL_IniProcDef( &C39_FTP_Transfer_to_MALDI_Reflex_III, "FTP-Transfer to MALDI Reflex III", 621, 0,
NULL, sizeof(MP2_PROC_CONTEXT_DEF) );
    MSL_IniProcDef( &C3A_User_Information, "User Information", 623, 0, NULL, sizeof(MP2_PROC_CONTEXT_DEF)
);

    return 0;
}

/*****
*
*   Labware Definitions
*
*****/

int IniAllLabwareDefs()
{
    IniLabwareDef( "C:\\BIOCHEM\\Packard\\MultiPROBE\\bin\\Robert\\Method Packages\\MALDI-LS 1.4s\\MALDI-
LS 1.41s\\MALDI-LS 1.42s_Labware.csv" );

    return 0;
}

/*****
*
*   User Defined Functions
*
*****/

//-----
// Uf_dDspVolume_S1()
//-----

int Uf_dDspVolume_S1(    // 0=Normal; 3=Abort
    char* pPCX ) // Address of procedure context information

```

```
{
    int nRet = 0; // Load return value into nRet
    MP2_PROC_CONTEXT_DEF* pPC = pPCX; // Cast pPCX into local procedure context ptr

// Load the variable Rt_dDspVolume_S1 with the desired value.
// If you set nRet to 3, the script will be aborted.
// Calculation of enzyme #1 volume to dispense in each Vial of
// the heating unit
// This calculation assumes, that you dispense one type of enzyme
// not more than once per sample. Otherwise you have to introduce
// a factor in the final volume calculation !

long Rec;

Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
Rec=Rec-3; // Subtracting headers and calibrant lines
Rt_dDspVolume_S1=(Rec/4)+10; // Calculation of dispense Volume per vial
// with an offset of 10ul

    return nRet;
}
// End of Uf_dDspVolume_S1()

//-----
// Uf_dDspVolume_S2()
//-----

int Uf_dDspVolume_S2( // 0=Normal; 3=Abort
    char* pPCX ) // Address of procedure context information
{
    int nRet = 0; // Load return value into nRet
    MP2_PROC_CONTEXT_DEF* pPC = pPCX; // Cast pPCX into local procedure context ptr

// Load the variable Rt_dDspVolume_S2 with the desired value.
// If you set nRet to 3, the script will be aborted.
// Calculation of enzyme # 2 volume to dispense in each Vial of
// the heating unit
// This calculation assumes, that you dispense one type of enzyme
// not more than once per sample. Otherwise you have to introduce
// a factor in the final volume calculation !

long Rec;

Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
Rec=Rec-3; // Subtracting headers and calibrant lines
Rt_dDspVolume_S2=(Rec/4)+10; // Calculation of dispense Volume per vial
// with an offset of 10ul

    return nRet;
}
// End of Uf_dDspVolume_S2()

//-----
// Uf_dDspVolume_S3()
//-----

int Uf_dDspVolume_S3( // 0=Normal; 3=Abort
    char* pPCX ) // Address of procedure context information
{
    int nRet = 0; // Load return value into nRet
    MP2_PROC_CONTEXT_DEF* pPC = pPCX; // Cast pPCX into local procedure context ptr

// Load the variable Rt_dDspVolume_S3 with the desired value.
// If you set nRet to 3, the script will be aborted.
// Calculation of enzyme #3 volume to dispense in each Vial of
// the heating unit
// This calculation assumes, that you dispense one type of enzyme
// not more than once per sample. Otherwise you have to introduce
// a factor in the final volume calculation !

long Rec;

Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
Rec=Rec-3; // Subtracting headers and calibrant lines
Rt_dDspVolume_S3=(Rec/4)+10; // Calculation of dispense Volume per vial
// with an offset of 10ul
```

```

    return nRet;
}
// End of Uf_dDspVolume_S3()

//-----
// Uf_stTimePeriod_S1()
//-----

int Uf_stTimePeriod_S1(    // 0=Normal; 3=Abort
    char* pPCX )  // Address of procedure context information
{
    int nRet = 0;                // Load return value into nRet
    MP2_PROC_CONTEXT_DEF* pPC = pPCX;  // Cast pPCX into local procedure context ptr

    // Load the variable Rt_stTimePeriod_S1 with the desired value.
    // If you set nRet to 3, the script will be aborted.
    // Drytime for the samples !

    long Rec, DryTimeT, DryTime;

    Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
    Rec=Rec-3;                                           // Subtracting headers and calibrant lines

    if(Rec<=10) DryTime=300;
    if(Rec>10 && Rec<=30) DryTimeT=30;
    if(Rec>30 && Rec<=50) DryTimeT=20;
    if(Rec>50) DryTimeT=0;

    if(Rec>10) DryTime=DryTimeT*Rec;                    // Calculation of the drytime, resulting from
    pipetting.                                           // It is estimated, that the pipetting of 4 samples
    to                                                  // the MALDI target takes about 15 seconds

    MSL_SecondsToTime( DryTime, &Rt_stTimePeriod_S1);

    return nRet;
}
// End of Uf_stTimePeriod_S1()

//-----
// Uf_stTimePeriod_S2()
//-----

int Uf_stTimePeriod_S2(    // 0=Normal; 3=Abort
    char* pPCX )  // Address of procedure context information
{
    int nRet = 0;                // Load return value into nRet
    MP2_PROC_CONTEXT_DEF* pPC = pPCX;  // Cast pPCX into local procedure context ptr

    // Load the variable Rt_stTimePeriod_S2 with the desired value.
    // If you set nRet to 3, the script will be aborted.
    // Drytime for the matrix on samples !

    long Rec, DryTimeT, DryTime;

    Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
    Rec=Rec-3;                                           // Subtracting headers and calibrant lines

    if(Rec<=10) DryTime=300;
    if(Rec>10 && Rec<=30) DryTimeT=30;
    if(Rec>30 && Rec<=50) DryTimeT=20;
    if(Rec>50) DryTimeT=0;

    if(Rec>10) DryTime=DryTimeT*Rec;                    // Calculation of the drytime, resulting from
    pipetting.                                           // It is estimated, that the pipetting of 4 samples
    to                                                  // the MALDI target takes about 15 seconds

```

```

MSL_SecondsToTime( DryTime, &Rt_stTimePeriod_S1);

    return nRet;
}
// End of Uf_stTimePeriod_S2()

//-----
// Uf_dDspVolume_S4()
//-----

int Uf_dDspVolume_S4(    // 0=Normal; 3=Abort
    char* pPCX )  // Address of procedure context information
{
    int nRet = 0;                // Load return value into nRet
    MP2_PROC_CONTEXT_DEF* pPC = pPCX;  // Cast pPCX into local procedure context ptr

    // Load the variable Rt_dDspVolume_S4 with the desired value.
    // If you set nRet to 3, the script will be aborted.
    // Calculation of Buffer #1 volume to dispense in each Vial
    // This calculation assumes, that you dispense the buffer
    // only once per sample.
    // Since the Buffer should not be limiting, its needed volume
    // is taken twice.

    long Rec;

    Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
    Rec=Rec-4;                // Subtracting headers and calibrant lines
    Rt_dDspVolume_S4=Rec+10;    // Calculation of dispense Volume per vial
                                // with an offset of 10ul

    return nRet;
}
// End of Uf_dDspVolume_S4()

//-----
// Uf_dDspVolume_S5()
//-----

int Uf_dDspVolume_S5(    // 0=Normal; 3=Abort
    char* pPCX )  // Address of procedure context information
{
    int nRet = 0;                // Load return value into nRet
    MP2_PROC_CONTEXT_DEF* pPC = pPCX;  // Cast pPCX into local procedure context ptr

    // Load the variable Rt_dDspVolume_S5 with the desired value.
    // If you set nRet to 3, the script will be aborted.
    // Calculation of Buffer #2 volume to dispense in each Vial
    // This calculation assumes, that you dispense the buffer
    // only once per sample.
    // Since the Buffer should not be limiting, its needed volume
    // is taken twice.

    long Rec;

    Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
    Rec=Rec-4;                // Subtracting headers and calibrant lines
    Rt_dDspVolume_S5=Rec+10;    // Calculation of dispense Volume per vial
                                // with an offset of 10ul

    return nRet;
}
// End of Uf_dDspVolume_S5()

//-----
// Uf_dDspVolume_S6()
//-----

int Uf_dDspVolume_S6(    // 0=Normal; 3=Abort
    char* pPCX )  // Address of procedure context information
{
    int nRet = 0;                // Load return value into nRet

```

```

    MP2_PROC_CONTEXT_DEF* pPC = pPCX;    // Cast pPCX into local procedure context ptr

// Load the variable Rt_dDspVolume_S6 with the desired value.
// If you set nRet to 3, the script will be aborted.
// Calculation of Buffer #3 volume to dispense in each Vial
// This calculation assumes, that you dispense the buffer
// only once per sample.
// Since the Buffer should not be limiting, its needed volume
// is taken twice.

long Rec;

Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
Rec=Rec-4;                                           // Subtracting headers and calibrant lines
Rt_dDspVolume_S6=Rec+10;                           // Calculation of dispense Volume per vial
                                                    // with an offset of 10ul

    return nRet;
}
// End of Uf_dDspVolume_S6()

//-----
// Uf_dDspVolume_S7()
//-----

int Uf_dDspVolume_S7(    // 0=Normal; 3=Abort
    char* pPCX ) // Address of procedure context information
{
    int nRet = 0;                // Load return value into nRet
    MP2_PROC_CONTEXT_DEF* pPC = pPCX;    // Cast pPCX into local procedure context ptr

// Load the variable Rt_dDspVolume_S7 with the desired value.
// If you set nRet to 3, the script will be aborted.
// Calculation of Buffer #4 volume to dispense in each Vial
// This calculation assumes, that you dispense the buffer
// only once per sample.
// Since the Buffer should not be limiting, its needed volume
// is taken twice.

long Rec;

Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
Rec=Rec-4;                                           // Subtracting headers and calibrant lines
Rt_dDspVolume_S7=Rec+10;                           // Calculation of dispense Volume per vial
                                                    // with an offset of 10ul

    return nRet;
}
// End of Uf_dDspVolume_S7()

//-----
// Uf_dDspVolume_S8()
//-----

int Uf_dDspVolume_S8(    // 0=Normal; 3=Abort
    char* pPCX ) // Address of procedure context information
{
    int nRet = 0;                // Load return value into nRet
    MP2_PROC_CONTEXT_DEF* pPC = pPCX;    // Cast pPCX into local procedure context ptr

// Load the variable Rt_dDspVolume_S8 with the desired value.
// If you set nRet to 3, the script will be aborted.
// Calculation of "DHB Matrix" volume to dispense in each Vial
// This calculation takes into account, that you dispense the matrix
// more than once per sample by looking if multiple matrix dispense.
// was chosen in the "GeneralTestInformation" Variable.
// The information about Multiple Matrix addition MUST stand in the
// first column, otherwise it will be ignored (even if you choose
// the right column). MSL_FileGetString or MSL FileGetLong do not
// work properly !

long m, a, Rec;

Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file

```

```

Rec=Rec-4; // Subtracting headers and calibrant lines
m=MSL_GetFileLong(Rt_GeneralTestInformation, 2, 0); // Checks if MultiAddition is desired or not
if (m==1) a=2; // Multi Addition is desired
else a=1; // Multi Addition is not desired
Rt_dDspVolume_S8=a*Rec+20; // Calculation of dispense Volume per vial
// with an offset of 20ul

return nRet;
}
// End of Uf_dDspVolume_S8()

//-----
// Uf_dDspVolume_S9()
//-----

int Uf_dDspVolume_S9( // 0=Normal; 3=Abort
char* pPCX ) // Address of procedure context information
{
int nRet = 0; // Load return value into nRet
MP2_PROC_CONTEXT_DEF* pPC = pPCX; // Cast pPCX into local procedure context ptr

// Load the variable Rt_dDspVolume_S9 with the desired value.
// If you set nRet to 3, the script will be aborted.
// Calculation of "alpha Matrix" volume to dispense in each Vial
// This calculation takes into account, that you dispense the matrix
// more than once per sample by looking if multiple matrix dispense.
// was choosen in the "GeneralTestInformation" Variable.
// The information about Multiple Matrix addition MUST stand in the
// first column, otherwise it willbe ignored (even if you choose
// the right column). MSL_FileGetString or MSL FileGetLong do not
// work properly !

long m, a, Rec;

Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
Rec=Rec-4; // Subtracting headers and calibrant lines
m=MSL_GetFileLong(Rt_GeneralTestInformation, 2, 0); // Checks if MultiAddition is desired or not
if (m==1) a=2; // Multi Addition is desired
else a=1; // Multi Addition is not desired
Rt_dDspVolume_S9=a*Rec+20; // Calculation of dispense Volume per vial
// with an offset of 20ul

return nRet;
}
// End of Uf_dDspVolume_S9()

//-----
// Uf_dDspVolume_S10()
//-----

int Uf_dDspVolume_S10( // 0=Normal; 3=Abort
char* pPCX ) // Address of procedure context information
{
int nRet = 0; // Load return value into nRet
MP2_PROC_CONTEXT_DEF* pPC = pPCX; // Cast pPCX into local procedure context ptr

// Load the variable Rt_dDspVolume_S10 with the desired value.
// If you set nRet to 3, the script will be aborted.
// Calculation of the Reagent volume to dispense in each Vial
// This calculation assumes, that you dispense the buffer
// only once per sample.
// Since the Reagent should not be limiting, its needed volume
// is taken twice. This seems also to be neccesary, since many
// Reagents are solumilized in volatile buffers.

long Rec;

Rec=MSL_GetFileLines(Rt_ExternalDataFileName, NULL); // Calculation of samples from file
Rec=Rec-4; // Subtracting headers and calibrant lines
Rt_dDspVolume_S10=Rec+10; // Calculation of dispense Volume per vial
// with an offset of 10ul

return nRet;
}
// End of Uf_dDspVolume_S10()

```

```

/*****
*
*   Procedure Functions...
*
*   Each procedure name is prefixed with a 'P' followed by one or
*   more pairs of hexadecimal digits. The number of pairs indicates
*   the nesting level of the procedure node in the test outline.
*   The value of each pair is the offset of the node at each level
*   from its parent.
*
*****/

/*****
*
*   P00_Verify_Deck_Layout()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P00_Verify_Deck_Layout()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C00_Verify_Deck_Layout;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    int nDlg;                // Dialog Id
    int nCurPage;           // Current Dialog Page Number
    int nNxtPage;           // Next Dialog Page Number
    int nX, nY;              // Control origin
    int nW, nH;              // Control size
    int nMaxY = 0;          // Max Y position
    MP2_DATE stMinDate;      // Minimum Date
    MP2_DATE stMaxDate;      // Maximum Date
    MP2_TIME stMinTime;      // Minimum Time
    MP2_TIME stMaxTime;      // Maximum Time

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Create the dialog box.
    nDlg = MSL_GetAvailableDialogId();
    MSL_CreateDialog( 0, nDlg, -1, -1, -1, -1,
        "Verify Deck-Layout", 0 );

    // Create page 1: Verify & Reset Btns.
    MSL_CreateLabel( nDlg, 6, 1, 1, 30, 1, "Test Startup Page", 0 );
    MSL_CreatePushBtn( nDlg, 7, 3, 3, 20, 1, "&Verify Labware Locations...", "Display the 'Verify Labware Locations' dialog.", 0, 0, 0 );
    MSL_CreatePushBtn( nDlg, 8, 3, -1, 20, 1, "&Reset Tip Boxes...", "Display the 'Reset Tip Boxes' dialog.", 0, 0, 0 );
    MSL_GetControlGeom( nDlg, 8, &nX, &nY, &nW, &nH );
    if( nMaxY < (nY + nH) ) nMaxY = nY + nH;
    MSL_CreateCollection( nDlg, 1, "6,7,8", 0, 0 );
    MSL_ShowControl( nDlg, 1, 0 );

    // Create page 2: Label, prompts and collection.
    MSL_CreateLabel( nDlg, 10, 1, 1, 26, 1, "General Test Information ?", 0 );
    MSL_CreateEditBox( nDlg, 11, -308, 3, 15, 1, "Browse for General Test Infomation", "Browse for the File Containing the General Test Information", &Rt_GeneralTestInformation, "TEXT", 0., 255., 0., 0, 0x0 );

    MSL_GetControlGeom( nDlg, 11, &nX, &nY, &nW, &nH );
    MSL_CreatePushBtn( nDlg, 12, -1, -nY, 3, 1, "...", "Browse for the File Containing the General Test Information", 0, 0, 0 );
    MSL_BtnSetSelectFile( nDlg, 12, 11, "File Containing the General Test Information ?", "Text Files - Tabulator Delimited (*.txt)|*.txt|Initialization Files (*.ini)|*.ini|All Files (*.*)|*.*||", "*.txt", "C:\\packard\\Multiprobe\\Robert\\Import Data\\queued samples", &Rt_GeneralTestInformation, 255, NULL, 0, NULL, 0x0 );
    MSL_GetControlGeom( nDlg, 12, &nX, &nY, &nW, &nH );
    if( nMaxY < (nY + nH) ) nMaxY = nY + nH;
    MSL_CreateCollection( nDlg, 2, "10,11,12", 0, 0 );
    MSL_ShowControl( nDlg, 2, 0 );
}

```



```

// Create page 3: Label, prompts and collection.
MSL_CreateLabel( nDlg, 13, 1, 1, 17, 1, "Stock Solutions ?", 0 );
MSL_CreateEditBox( nDlg, 14, -308, 3, 15, 1, "File containing information about stock solution
sources:", "", &Rt_StockSolSource, "TEXT", 0., 255., 0., 0, 0x0 );
MSL_GetControlGeom( nDlg, 14, &nX, &nY, &nW, &nH );
MSL_CreatePushBtn( nDlg, 15, -1, -nY, 3, 1, "...", "", 0, 0, 0 );
MSL_BtnSetSelectFile( nDlg, 15, 14, "Browse for stock solution source file", "Text Files - Tabulator
Delimited (*.txt)|*.txt|All Files (*.*)|*.*||", "txt", "C:\\packard\\Multiprobe\\Robert\\Import
Data\\queued samples", &Rt_StockSolSource, 255, NULL, 0, NULL, 0x0 );
MSL_CreateEditBox( nDlg, 16, -308, -1, 15, 1, "File containing information about stock solution
destinations:", "", &Rt_StockSolDest, "TEXT", 0., 255., 0., 0, 0x0 );
MSL_GetControlGeom( nDlg, 16, &nX, &nY, &nW, &nH );
MSL_CreatePushBtn( nDlg, 17, -1, -nY, 3, 1, "...", "", 0, 0, 0 );
MSL_BtnSetSelectFile( nDlg, 17, 16, "Browse for stock solution destinations file", "Text Files -
Tabulator Delimited (*.txt)|*.txt|All Files (*.*)|*.*||", "txt", "C:\\packard\\Multiprobe\\Robert\\Import
Data\\queued samples", &Rt_StockSolDest, 255, NULL, 0, NULL, 0x0 );
MSL_GetControlGeom( nDlg, 17, &nX, &nY, &nW, &nH );
if( nMaxY < (nY + nH) ) nMaxY = nY + nH;
MSL_CreateCollection( nDlg, 3, "13,14,15,16,17", 0, 0 );
MSL_ShowControl( nDlg, 3, 0 );

// Create page 4: Label, prompts and collection.
MSL_CreateLabel( nDlg, 18, 1, 1, 25, 1, "Location of Sample Data ?", 0 );
MSL_CreateEditBox( nDlg, 19, -308, 3, 15, 1, "Browse for External Sample Data", "Browse for the File
Containing the External Sample Data", &Rt_ExternalDataFileName, "TEXT", 0., 255., 0., 0, 0x0 );
MSL_GetControlGeom( nDlg, 19, &nX, &nY, &nW, &nH );
MSL_CreatePushBtn( nDlg, 20, -1, -nY, 3, 1, "...", "Browse for the File Containing the External Sample
Data", 0, 0, 0 );
MSL_BtnSetSelectFile( nDlg, 20, 19, "Browse for External Sample Data", "Text Files - Tabulator
Delimited (*.txt)|*.txt|External Worklist (*.csv)|*.csv|Multiprobe Worklist (*.wkl)|*.wkl|All Files
(*.*)|*.*||", "txt", "C:\\packard\\Multiprobe\\Robert\\Import Data\\queued samples",
&Rt_ExternalDataFileName, 255, NULL, 0, NULL, 0x0 );
MSL_GetControlGeom( nDlg, 20, &nX, &nY, &nW, &nH );
if( nMaxY < (nY + nH) ) nMaxY = nY + nH;
MSL_CreateCollection( nDlg, 4, "18,19,20", 0, 0 );
MSL_ShowControl( nDlg, 4, 0 );

// Create page 5: Label, prompts and collection.
MSL_CreateLabel( nDlg, 21, 1, 1, 28, 1, "Location of Calibrant Data ?", 0 );
MSL_CreateEditBox( nDlg, 22, -308, 3, 15, 1, "Browse for Calibrant Data", "Browse for the File
Containing the Calibrant Data", &Rt_Calibrant, "TEXT", 0., 255., 0., 0, 0x0 );
MSL_GetControlGeom( nDlg, 22, &nX, &nY, &nW, &nH );
MSL_CreatePushBtn( nDlg, 23, -1, -nY, 3, 1, "...", "Browse for the File Containing the Calibrant
Data", 0, 0, 0 );
MSL_BtnSetSelectFile( nDlg, 23, 22, "Browse for External Calibrant Data", "Text Files - Tabulator
Delimited (*.txt)|*.txt|External Worklist (*.csv)|*.csv|Multiprobe Worklist (*.wkl)|*.wkl|All Files
(*.*)|*.*||", "txt", "C:\\packard\\Multiprobe\\Robert\\Import Data\\queued samples", &Rt_Calibrant, 255,
NULL, 0, NULL, 0x0 );
MSL_GetControlGeom( nDlg, 23, &nX, &nY, &nW, &nH );
if( nMaxY < (nY + nH) ) nMaxY = nY + nH;
MSL_CreateCollection( nDlg, 5, "21,22,23", 0, 0 );
MSL_ShowControl( nDlg, 5, 0 );

// Create push buttons...
nMaxY += 28;
MSL_CreatePushBtn( nDlg, 10001, 1, -nMaxY, 10, 1, "< &Back", "Display Previous Page", 0, 0, 0 );
MSL_CreatePushBtn( nDlg, 10002, -1, -nMaxY, 10, 1, "&Next >", "Display Next Page", 0, 0, 0 );
MSL_CreatePushBtn( nDlg, 10003, 24, -nMaxY, 10, 1, "&Start", "Begin Test Execution", 0, 0, 0x10 );
MSL_CreatePushBtn( nDlg, 10004, -1, -nMaxY, 10, 1, "&Abort", "Quit Test Now", 0, 0, 0 );

// Procedure query loop.
pPC->nProcLimit = 1;
for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Increment the loop count.
    nLoopCount++;
}

```

```

// Reset the current procedure context.
MSL_SetCurrentProcContext( pPC, 1, 0 );

// Sound the buzzer for 1/8 second
MSL_Beep( 500, 0.125 );

// Display the dialog box.
nCurPage = 0;
nNxtPage = 1;
MSL_ShowDialog( nDlg, 1, 0 );

// Monitor user button selections...
while( 1 )
{
    // Was there a change page request?
    if( nCurPage != nNxtPage )
    {
        if( nCurPage > 0 )    // Hide current page (if any)
        {
            MSL_ShowControl( nDlg, nCurPage, 0 );    // Hide current page
        }
        if( nNxtPage > 0 )    // Show next page (if any)
        {
            MSL_ShowControl( nDlg, nNxtPage, 1 );    // Show next page
            MSL_UpdateControl( nDlg, nNxtPage );    // Update control values on next page
            MSL_SetControlFocus( nDlg, nNxtPage );    // Give Focus to first control on next page
        }
        else break;    // Quit if next page not > 0

        nCurPage = nNxtPage;
    }

    // Set button states according to the current page.
    MSL_EnableControl( nDlg, 10001, (nCurPage > 1) );
    MSL_EnableControl( nDlg, 10002, (nCurPage < 5) );
    MSL_ShowControl( nDlg, 10003, (nCurPage == 5) );

    // Wait for the user to click a button.
    nRet = MSL_WaitControlList( nDlg, "10001,10002,10003,10004,7,8", 0, -1, 0 );

    switch( nRet )
    {
        case 7:    // Verify Labware
            MSL_VerifyLabwareDialog();
            break;

        case 8:    // Reset Tip Boxes
            MSL_ResetTipBoxesDialog();
            break;

        case 10001:    // Back
            nNxtPage = nCurPage - 1;
            if( nNxtPage < 1 ) nNxtPage = 1;
            break;

        case 10002:    // Next
            nNxtPage = nCurPage + 1;
            if( nNxtPage > 5 ) nNxtPage = 5;
            break;

        case 10003:    // Start
            nNxtPage = 0;
            break;

        case 10004:    // Abort?
            MSL_DeleteDialog( nDlg, 0 );
            MSL_Abort();
            return -1;
    }
}

} //<End Monitor button selections>

// Hide the dialog box.
MSL_ShowDialog( nDlg, 0, 0 );

} //<End Procedure Loop>

// Delete the dialog box.

```

```

MSL_DeleteDialog( nDlg, 0 );

// Remove disposable tips (if any)
MSL_RemovedTs();

return 0;
} //<End P00_Verify_Deck_Layout()>

/*****
*
*   P01_Danger_of_Test_Failiure_and_LVT_Damage__()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P01_Danger_of_Test_Failiure_and_LVT_Damage__()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C01_Danger_of_Test_Failiure_and_LVT_Damage__;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Sound the buzzer for 1/8 second
        MSL_Beep( 500, 0.125 );

        // Display User Message.
        nRet = MSL_MessageDialog( 0, // No dialog owner
                                "Danger of Test Failiure and LVT-Damage !", // Dialog Title
                                "Remove Caps from all used Solutions !" );

        Check List:

        -> Diluent-Solution Trough
        -> Organic Flush/Wash Troughs
        -> Enzyme-Solution Microfuges
        -> Calibrant-Solution Microfuges
        -> Matrix-Solution Microfuges
        ", // Dialog Message
          2, // Icon: Exclamation
          1, // Btns: OK
          1, // Default to 1st button
          0 ); // Reserved flags field

        // Test User Reply: 2=>Cancel, 7=>No
        if( (nRet == 2) || (nRet == 7) ) return -1;
    }
}

```

```

    } //<End Procedure Loop>

    return 0;

} //<End P01_Danger_of_Test_Failiure_and_LVT_Damage__()>

/*****
*
*   P02_Pre_Test_Flush_Wash()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P02_Pre_Test_Flush_Wash()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C02_Pre_Test_Flush_Wash;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    int nMask;          // General purpose mask

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 1 );

        // Perform step 'Pre-Test Flush' for all dilutors
        nRet = MSL_WashAllDilutorsInProc(
            pPC, 6,
            &M0200_Pre_Test_Flush_1,
            1, // Pump: 1=Peri, 0=Syringe
            3000.0, // System liquid volume
            0.0, // post-flush system air gap
            1, // wash cycles
            1, 1.0, 0.0, -1, 0, 10.0, -1, 0,
            "", // pre-step callback
            "" ); // post-step callback
        if( nRet == -1 ) return nRet;

        // Perform step 'Pre-Test Wash' for all dilutors
        nRet = MSL_WashAllDilutorsInProc(
            pPC, 9,
            &M0201_Pre_Test_Wash_1,
            1, // Pump: 1=Peri, 0=Syringe
            3000.0, // System liquid volume
            5.0, // post-flush system air gap
            1, // wash cycles
            1, 1.0, 0.0, -1, 0, 10.0, -1, 0,

```

```

        "",      // pre-step callback
        "" );    // post-step callback
    if( nRet == -1 ) return nRet;

} //<End Procedure Loop>

return 0;

} //<End P02_Pre_Test_Flush_Wash()>

/*****
*
*   P0300_Enzyme__1_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0300_Enzyme__1_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0300_Enzyme__1_Panel;
    MP2_PROC_CONTEXT_DEF *pParentPC = pPC->pParentPC;
    int nRet;                // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    int bEOM = 0;            // End of map indicator
    int nXfrGroups;          // Nbr of transfer groups constructed for the sample
    int nMask;               // General purpose mask
    double dMixVol;           // Mix volume (µL)
    double dAspVol;           // Aspirate volume (µL)
    double dDspVol;           // Dispense volume (µL)
    double dXfrVol;           // Total transfer volume (µL)
    double dWasteVol = 0;     // Waste volume (µL)
    double dBlowoutVol = 0;   // Blowout volume (µL)
    double dPostAir;          // Post-step aspirate air gap (µL)
    int nDsp;                 // Dispense counter
    int nDspMax;              // Maximum number of dispenses
    int nDspMode;             // Dispense mode
    double dLLSAspHeight;     // Liq. Level Sense Aspirate Height
    double dLLSDspHeight;     // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed
    int nGrpWells;            // Nbr of wells to dispense in xfr group
    int nDspWells;           // Remaining wells to dispense in xfr group

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );
    EGS_SetLiquidType( "SyringeTest.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Start with caller specified sequence count.
    pPC->nSeqCnt = pParentPC->nSeqCnt;

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-N-N-N" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = 1;
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;

```

```

for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 1, "E1", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    pParentPC->nSeqCnt = pPC->nSeqCnt;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 2 ); // Waste Mode

    //-----
    // Transfer Group_1
    //-----
    nGrpWells = 4;
    nDspWells = nGrpWells;
    MSL_InitXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
    MSL_SetAspStepVol( 0, 1, &M03000000_Aspirate, 0.0 );
    MSL_SetDspStepVol( 0, 1, &M03000001_Enzyme_1_Dispense, MSL_CallFloatVarUf( "Uf_dDspVolume_S1",
&Rt_dDspVolume_S1, pPC ), 4 );
    if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
    {
        dXfrVol = MSL_CalAspStepVol( nDspWells );

        //-----
        // Aspirate
        //-----
        if( MSL_UseAspStep( 0, 1 ) )
        {
            dAspVol = MSL_GetAspStepVol( 0 );

            if( nDspWells == nGrpWells )
                dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
            else
                dWasteVol = 0;

            dDspVol = 0.0;

            nRet = EGS_AspEx( pPC, 16, &M03000000_Aspirate, 0,
                                "E1_P", // sample id
                                dAspVol, // aspirate volume
                                0.0, // pre-aspirate air volume (for blowout)
                                3.0, // post-aspirate air volume
                                dDspVol, // dispense back volume
                                dWasteVol ); // Waste volume
            if( nRet == -1 ) return nRet;
            if( nRet == -2 ) { bEOM = 1; break; }
            if( nRet == -3 ) break; // Skip Sample? // Specify additional aspirate
        }
    }
}

```

details...

```

        EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
        PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
        dLLSCLotDetHeight, 1 );
    }

    //-----
    // Enzyme 1:Dispense
    //-----
    if( MSL_UseDspStep( 0, 1, &nDspWells ) )
    {

        dDspVol = MSL_GetDspStepVol( 0 );

        dPostAir = 3.0;
        nDspMode = 0;

        nDspMax = 4;    // Load maximum dispenses
        for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
        {
            nRet = EGS_DspEx( pPC, 17, &M03000001_Enzyme_1_Dispense, 1,
                            dDspVol,    // dispense volume
                            dPostAir,    // post air volume
                            nDspMode,    // dispense mode
                            0 );        // Reserved for blowout delay
            if( nRet == -1 ) return nRet;
            if( nRet == -2 ) { bEOM = 1; break; }
            nDspWells--;
            // Specify additional dispense details...
            EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
            PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
        }

        // Increment the transfer group count.
        nXfrGroups++;

        // Break out of transfer group of EOM reached.
        if( bEOM ) break;
    }

    // Clear the sequence if a skip sample was returned
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

    // Quit if EOM reached and no wells were dispensed
    if( bEOM && (nDspWells == nGrpWells) ) break;

    //-----
    // Flush_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 19, &M030001_Flush_1, 0,
                        dDspVol,    // system liquid volume
                        0.0,        // post-wash system air gap
                        1 );        // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // Wash_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 22, &M030002_Wash_1, 0,
                        dDspVol,    // system liquid volume
                        5.0,        // post-wash system air gap
                        1 );        // wash cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Backout the current sequence if no transfer groups were created.
    if( nXfrGroups == 0 )
    {

```

```

        EGS_ClearCurSeq( pPC );
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1;    // Abort the test.
    if( nRet == 2 ) return 0;    // Abort this procedure but not the test.
    pParentPC->nSeqCnt = pPC->nSeqCnt;

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

return 0;

} //<End P0300_Enzyme__1_Panel()>

/*****
*
*   P0301_Enzyme__2_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0301_Enzyme__2_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0301_Enzyme__2_Panel;
    MP2_PROC_CONTEXT_DEF *pParentPC = pPC->pParentPC;
    int nRet;                // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;              // Last procedure loop flag
    int bEOM = 0;           // End of map indicator
    int nXfrGroups;         // Nbr of transfer groups constructed for the sample
    int nMask;              // General purpose mask
    double dMixVol;         // Mix volume (µL)
    double dAspVol;         // Aspirate volume (µL)
    double dDspVol;         // Dispense volume (µL)
    double dXfrVol;         // Total transfer volume (µL)
    double dWasteVol = 0;    // Waste volume (µL)
    double dBlowoutVol = 0;  // Blowout volume (µL)
    double dPostAir;        // Post-step aspirate air gap (µL)
    int nDsp;               // Dispense counter
    int nDspMax;            // Maximum number of dispenses
    int nDspMode;           // Dispense mode
    double dLLSAspHeight;   // Liq. Level Sense Aspirate Height
    double dLLSDspHeight;   // Liq. Level Sense Dispense Height
    double dLLSclotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed
    int nGrpWells;          // Nbr of wells to dispense in xfr group
    int nDspWells;          // Remaining wells to dispense in xfr group

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );
    EGS_SetLiquidType( "SyringeTest.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSclotDetHeight = PrfFileGetLLSclotDetHeight( pPC->hPrfFile );

```



```

dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Start with caller specified sequence count.
pPC->nSeqCnt = pParentPC->nSeqCnt;

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "N-Y-N-N" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 2, "E2", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    pParentPC->nSeqCnt = pPC->nSeqCnt;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 2 ); // Waste Mode

    //-----
    // Transfer Group_1
    //-----
    nGrpWells = 4;
    nDspWells = nGrpWells;
    MSL_IniXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
    MSL_SetAspStepVol( 0, 1, &M03010000_Aspirate, 0.0 );
    MSL_SetDspStepVol( 0, 1, &M03010001_Dspl_Dispense, MSL_CallFloatVarUf( "Uf_dDspVolume_S2",
&Rt_dDspVolume_S2, pPC ), 4 );
    if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
    {
        dXfrVol = MSL_CalAspStepVol( nDspWells );

        //-----
        // Aspirate
        //-----
        if( MSL_UseAspStep( 0, 1 ) )
        {
            dAspVol = MSL_GetAspStepVol( 0 );

            if( nDspWells == nGrpWells )
                dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
            else
                dWasteVol = 0;

```

```

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 28, &M03010000_Aspirate, 0,
                     "E2_P",          // sample id
                     dAspVol,         // aspirate volume
                     0.0,             // pre-aspirate air volume (for blowout)
                     3.0,             // post-aspirate air volume
                     dDspVol,         // dispense back volume
                     dWasteVol );     // Waste volume

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) break;          // Skip Sample?           // Specify additional aspirate
details...
    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
    PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
    dLLSCLotDetHeight, 1 );
}

//-----
// Dsp1:Dispense
//-----
if( MSL_UseDspStep( 0, 1, &nDspWells ) )
{
    dDspVol = MSL_GetDspStepVol( 0 );

    dPostAir = 3.0;
    nDspMode = 0;

    nDspMax = 4;    // Load maximum dispenses
    for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
    {
        nRet = EGS_DspEx( pPC, 29, &M03010001_Dsp1_Dispense, 1,
                          dDspVol,          // dispense volume
                          dPostAir,         // post air volume
                          nDspMode,        // dispense mode
                          0 );              // Reserved for blowout delay

        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        nDspWells--;
        // Specify additional dispense details...
        EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
    PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
    }

    // Increment the transfer group count.
    nXfrGroups++;

    // Break out of transfer group of EOM reached.
    if( bEOM ) break;
}

// Clear the sequence if a skip sample was returned
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

// Quit if EOM reached and no wells were dispensed
if( bEOM && (nDspWells == nGrpWells) ) break;

//-----
// Flush_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 31, &M030101_Flush_1, 0,
                      dDspVol,          // system liquid volume
                      0.0,             // post-wash system air gap
                      1 );              // flush cycles

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Wash_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 34, &M030102_Wash_1, 0,
                      dDspVol,          // system liquid volume

```

```

                    5.0,      // post-wash system air gap
                    1 );      // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Backout the current sequence if no transfer groups were created.
if( nXfrGroups == 0 )
{
    EGS_ClearCurSeq( pPC );
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;      // Abort the test.
if( nRet == 2 ) return 0;      // Abort this procedure but not the test.
pParentPC->nSeqCnt = pPC->nSeqCnt;

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

return 0;

} //<End P0301_Enzyme__2_Panel()>

/*****
*
*   P0302_Enzyme__3_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0302_Enzyme__3_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0302_Enzyme__3_Panel;
    MP2_PROC_CONTEXT_DEF *pParentPC = pParentPC;
    int nRet;      // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;      // Last procedure loop flag
    int bEOM = 0;    // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask;      // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp;      // Dispense counter
    int nDspMax;   // Maximum number of dispenses
    int nDspMode;  // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSclotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed
    int nGrpWells; // Nbr of wells to dispense in xfr group
    int nDspWells; // Remaining wells to dispense in xfr group

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

```

```

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );
EGS_SetLiquidType( "SyringeTest.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Start with caller specified sequence count.
pPC->nSeqCnt = pParentPC->nSeqCnt;

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "N-N-Y-N" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 3, "E3", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    pParentPC->nSeqCnt = pPC->nSeqCnt;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 2 ); // Waste Mode

    //-----
    // Transfer Group_1
    //-----
    nGrpWells = 4;
    nDspWells = nGrpWells;
    MSL_IniXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
    MSL_SetAspStepVol( 0, 1, &M03020000_Aspirate, 0.0 );
    MSL_SetDspStepVol( 0, 1, &M03020001_Dsp1_Dispense, MSL_CallFloatVarUf( "Uf_dDspVolume_S3",
&Rt_dDspVolume_S3, pPC ), 4 );
    if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
    {
        dxfrVol = MSL_CalAspStepVol( nDspWells );

        //-----

```

```

// Aspirate
//-----
if( MSL_UseAspStep( 0, 1 ) )
{
    dAspVol = MSL_GetAspStepVol( 0 );

    if( nDspWells == nGrpWells )
        dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
    else
        dWasteVol = 0;

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 40, &M03020000_Aspirate, 0,
                     "E3_P",          // sample id
                     dAspVol,         // aspirate volume
                     0.0,             // pre-aspirate air volume (for blowout)
                     3.0,             // post-aspirate air volume
                     dDspVol,         // dispense back volume
                     dWasteVol );     // Waste volume
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) break;          // Skip Sample?           // Specify additional aspirate
details...
    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
    PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
    dLLSCLotDetHeight, 1 );
}

//-----
// Dsp1:Dispense
//-----
if( MSL_UseDspStep( 0, 1, &nDspWells ) )
{
    dDspVol = MSL_GetDspStepVol( 0 );

    dPostAir = 3.0;
    nDspMode = 0;

    nDspMax = 4;    // Load maximum dispenses
    for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
    {
        nRet = EGS_DspEx( pPC, 41, &M03020001_Dsp1_Dispense, 1,
                         dDspVol,         // dispense volume
                         dPostAir,        // post air volume
                         nDspMode,        // dispense mode
                         0 );             // Reserved for blowout delay
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        nDspWells--;
        // Specify additional dispense details...
        EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
        PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
    }

    // Increment the transfer group count.
    nXfrGroups++;

    // Break out of transfer group if EOM reached.
    if( bEOM ) break;
}

// Clear the sequence if a skip sample was returned
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

// Quit if EOM reached and no wells were dispensed
if( bEOM && (nDspWells == nGrpWells) ) break;

//-----
// Flush_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 43, &M030201_Flush_1, 0,
                      dDspVol,         // system liquid volume
                      0.0,             // post-wash system air gap
                      1 );             // flush cycles
if( nRet == -1 ) return nRet;

```

```

    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // Wash_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 46, &M030202_Wash_1, 0,
                          dDspVol, // system liquid volume
                          5.0, // post-wash system air gap
                          1 ); // wash cycles

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Backout the current sequence if no transfer groups were created.
    if( nXfrGroups == 0 )
    {
        EGS_ClearCurSeq( pPC );
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1; // Abort the test.
    if( nRet == 2 ) return 0; // Abort this procedure but not the test.
    pParentPC->nSeqCnt = pPC->nSeqCnt;

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

return 0;
} //<End P0302_Enzyme__3_Panel()>

/*****
 *
 * P03_Enzyme_Panel()
 *
 * <Reserved for text entered into the procedure's comment page.>
 *
 *****/

int P03_Enzyme_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C03_Enzyme_Panel;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = 1;

```

```

pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    //-----
    // Enzyme #1 Panel
    //-----
    nRet = MSL_CallChildProc( "P0300_Enzyme__1_Panel", nLoopCount, 0, 1, bDone );
    if( nRet == -1 ) return -1;

    //-----
    // Enzyme #2 Panel
    //-----
    nRet = MSL_CallChildProc( "P0301_Enzyme__2_Panel", nLoopCount, 0, 1, bDone );
    if( nRet == -1 ) return -1;

    //-----
    // Enzyme #3 Panel
    //-----
    nRet = MSL_CallChildProc( "P0302_Enzyme__3_Panel", nLoopCount, 0, 1, bDone );
    if( nRet == -1 ) return -1;

    // If done, break now.
    if( bDone ) break;

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 0 );

} //<End Procedure Loop>

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1; // Abort the test.

return 0;

} //<End P03_Enzyme_Panel()>

/*****
*
*   P04_Enzyme_Pre_Incubation()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P04_Enzyme_Pre_Incubation()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C04_Enzyme_Pre_Incubation;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.

```

```

    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 0 );

    // Create/Update the Time Marker
    MSL_TimeMarkerCreate( pPC->szName );

} //<End Procedure Loop>

return 0;

} //<End P04_Enzyme_Pre_Incubation()>

/*****
*
*   P0500_Flush_Wash()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0500_Flush_Wash()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0500_Flush_Wash;
    MP2_PROC_CONTEXT_DEF *pParentPC = pPC->pParentPC;
    int nRet; // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int nMask; // General purpose mask

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Start with caller specified sequence count.
    pPC->nSeqCnt = pParentPC->nSeqCnt;

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 1 );

        // Perform step 'Flush' for all dilutors
        nRet = MSL_WashAllDilutorsInProc(
            pPC, 55,
            &M050000_Flush_1,
            1, // Pump: 1=Peri, 0=Syringe

```



```

        3000.0,      // System liquid volume
        0.0,        // post-flush system air gap
        1,          // wash cycles
        1, 1.0, 0.0, -1, 0, 10.0, -1, 0,
        "",         // pre-step callback
        "" );       // post-step callback
    if( nRet == -1 ) return nRet;

    // Perform step 'Wash' for all dilutors
    nRet = MSL_WashAllDilutorsInProc(
        pPC, 58,
        &M050001_Wash_1,
        1,      // Pump: 1=Peri, 0=Syringe
        3000.0, // System liquid volume
        5.0,    // post-flush system air gap
        1,      // wash cycles
        1, 1.0, 0.0, -1, 0, 10.0, -1, 0,
        "",     // pre-step callback
        "" );   // post-step callback
    if( nRet == -1 ) return nRet;

} //<End Procedure Loop>

return 0;

} //<End P0500_Flush_Wash()>

/*****
*
*   P0501_Buffer__1_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0501_Buffer__1_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0501_Buffer__1_Panel;
    MP2_PROC_CONTEXT_DEF *pParentPC = pPC->pParentPC;
    int nRet; // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed
    int nGrpWells; // Nbr of wells to dispense in xfr group
    int nDspWells; // Remaining wells to dispense in xfr group

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pParentPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );
    EGS_SetLiquidType( "SyringeTest.prf" );

```

```

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Start with caller specified sequence count.
pPC->nSeqCnt = pParentPC->nSeqCnt;

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 6, "B1", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    pParentPC->nSeqCnt = pPC->nSeqCnt;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 2 ); // Waste Mode

    //-----
    // Transfer Group_1
    //-----
    nGrpWells = 4;
    nDspWells = nGrpWells;
    MSL_IniXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
    MSL_SetAspStepVol( 0, 1, &M05010000_Aspirate, 0.0 );
    MSL_SetDspStepVol( 0, 1, &M05010001_Dsp1_Dispense, MSL_CallFloatVarUf( "Uf_dDspVolume_S4",
&Rt_dDspVolume_S4, pPC ), 4 );
    if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
    {
        dXfrVol = MSL_CalAspStepVol( nDspWells );

        //-----
        // Aspirate
        //-----
        if( MSL_UseAspStep( 0, 1 ) )
        {
            dAspVol = MSL_GetAspStepVol( 0 );

            if( nDspWells == nGrpWells )

```

```

        dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
    else
        dWasteVol = 0;

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 64, &M05010000_Aspirate, 0,
        "Bl_P",          // sample id
        dAspVol,         // aspirate volume
        0.0,            // pre-aspirate air volume (for blowout)
        3.0,            // post-aspirate air volume
        dDspVol,         // dispense back volume
        dWasteVol );    // Waste volume
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) break;    // Skip Sample?                // Specify additional aspirate
details...
    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
    PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
    dLLSCLotDetHeight, 1 );
    }

    //-----
    // Dsp1:Dispense
    //-----
    if( MSL_UseDspStep( 0, 1, &nDspWells ) )
    {
        dDspVol = MSL_GetDspStepVol( 0 );

        dPostAir = 3.0;
        nDspMode = 0;

        nDspMax = 4;    // Load maximum dispenses
        for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
        {
            nRet = EGS_DspEx( pPC, 65, &M05010001_Dsp1_Dispense, 1,
                dDspVol,        // dispense volume
                dPostAir,       // post air volume
                nDspMode,       // dispense mode
                0 );           // Reserved for blowout delay
            if( nRet == -1 ) return nRet;
            if( nRet == -2 ) { bEOM = 1; break; }
            nDspWells--;
            // Specify additional dispense details...
            EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
    PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
            }
        }

        // Increment the transfer group count.
        nXfrGroups++;

        // Break out of transfer group of EOM reached.
        if( bEOM ) break;
    }

    // Clear the sequence if a skip sample was returned
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

    // Quit if EOM reached and no wells were dispensed
    if( bEOM && (nDspWells == nGrpWells) ) break;

    //-----
    // Flush_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 67, &M050101_Flush_1, 0,
        dDspVol,        // system liquid volume
        0.0,           // post-wash system air gap
        1 );           // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // Wash_1

```

```

//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 70, &M050102_Wash_1, 0,
                      dDspVol, // system liquid volume
                      5.0, // post-wash system air gap
                      1 ); // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Backout the current sequence if no transfer groups were created.
if( nXfrGroups == 0 )
{
    EGS_ClearCurSeq( pPC );
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1; // Abort the test.
if( nRet == 2 ) return 0; // Abort this procedure but not the test.
pParentPC->nSeqCnt = pPC->nSeqCnt;

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

return 0;

} //<End P0501_Buffer__1_Panel()>

/*****
*
* P0502_Buffer__2_Panel()
*
* <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0502_Buffer__2_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0502_Buffer__2_Panel;
    MP2_PROC_CONTEXT_DEF *pParentPC = pParentPC;
    int nRet; // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height

    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed
    int nGrpWells; // Nbr of wells to dispense in xfr group
    int nDspWells; // Remaining wells to dispense in xfr group

```

```

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );
EGS_SetLiquidType( "SyringeTest.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSclotDetHeight   = PrfFileGetLLSclotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Start with caller specified sequence count.
pPC->nSeqCnt = pParentPC->nSeqCnt;

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 7, "B2", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    pParentPC->nSeqCnt = pPC->nSeqCnt;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 2 ); // Waste Mode

    //-----
    // Transfer Group_1
    //-----
    nGrpWells = 4;
    nDspWells = nGrpWells;
    MSL_IniXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
    MSL_SetAspStepVol( 0, 1, &M05020000_Aspirate, 0.0 );
    MSL_SetDspStepVol( 0, 1, &M05020001_Dsp1_Dispense, MSL_CallFloatVarUf( "Uf_dDspVolume_S5",
&Rt_dDspVolume_S5, pPC ), 4 );

```

```

if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
{
    dXfrVol = MSL_CalAspStepVol( nDspWells );

    //-----
    // Aspirate
    //-----
    if( MSL_UseAspStep( 0, 1 ) )
    {
        dAspVol = MSL_GetAspStepVol( 0 );

        if( nDspWells == nGrpWells )
            dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
        else
            dWasteVol = 0;

        dDspVol = 0.0;

        nRet = EGS_AspEx( pPC, 76, &M05020000_Aspirate, 0,
                          "B2_P",          // sample id
                          dAspVol,         // aspirate volume
                          0.0,             // pre-aspirate air volume (for blowout)
                          3.0,             // post-aspirate air volume
                          dDspVol,         // dispense back volume
                          dWasteVol );     // Waste volume
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        if( nRet == -3 ) break;           // Skip Sample?                // Specify additional aspirate
details...
        EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
        PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
        dLLSCLotDetHeight, 1 );
    }

    //-----
    // Dsp1:Dispense
    //-----
    if( MSL_UseDspStep( 0, 1, &nDspWells ) )
    {
        dDspVol = MSL_GetDspStepVol( 0 );

        dPostAir = 3.0;
        nDspMode = 0;

        nDspMax = 4;    // Load maximum dispenses
        for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
        {
            nRet = EGS_DspEx( pPC, 77, &M05020001_Dsp1_Dispense, 1,
                              dDspVol,         // dispense volume
                              dPostAir,        // post air volume
                              nDspMode,        // dispense mode
                              0 );            // Reserved for blowout delay
            if( nRet == -1 ) return nRet;
            if( nRet == -2 ) { bEOM = 1; break; }
            nDspWells--;
            // Specify additional dispense details...
            EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
            PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
        }

        // Increment the transfer group count.
        nXfrGroups++;

        // Break out of transfer group of EOM reached.
        if( bEOM ) break;
    }

    // Clear the sequence if a skip sample was returned
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

    // Quit if EOM reached and no wells were dispensed
    if( bEOM && (nDspWells == nGrpWells) ) break;

    //-----
    // Flush_1
    //-----

```

```

dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 79, &M050201_Flush_1, 0,
                      dDspVol,      // system liquid volume
                      0.0,          // post-wash system air gap
                      1 );          // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Wash_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 82, &M050202_Wash_1, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Backout the current sequence if no transfer groups were created.
if( nXfrGroups == 0 )
{
    EGS_ClearCurSeq( pPC );
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1; // Abort the test.
if( nRet == 2 ) return 0;   // Abort this procedure but not the test.
pParentPC->nSeqCnt = pPC->nSeqCnt;

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

return 0;

} //<End P0502_Buffer__2_Panel()>

/*****
*
*   P0503_Buffer__3_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0503_Buffer__3_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0503_Buffer__3_Panel;
    MP2_PROC_CONTEXT_DEF *pParentPC = pParentPC->pParentPC;
    int nRet; // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)

```

```

double dBlowoutVol = 0; // Blowout volume (µL)
double dPostAir;       // Post-step aspirate air gap (µL)
int nDsp;              // Dispense counter
int nDspMax;           // Maximum number of dispenses
int nDspMode;          // Dispense mode
double dLLSAspHeight;  // Liq. Level Sense Aspirate Height
double dLLSDspHeight;  // Liq. Level Sense Dispense Height
double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
double dRetractFromLiqHeight; // Retract from liquid height
double dRetractFromLiqSpeed; // Retract from liquid speed
int nGrpWells;         // Nbr of wells to dispense in xfr group
int nDspWells;         // Remaining wells to dispense in xfr group

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );
EGS_SetLiquidType( "SyringeTest.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Start with caller specified sequence count.
pPC->nSeqCnt = pParentPC->nSeqCnt;

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 8, "B3", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    pParentPC->nSeqCnt = pPC->nSeqCnt;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

```



```

// Load the pipette rule.
EGS_SetPipetteRule( 2 ); // Waste Mode

//-----
// Transfer Group_1
//-----
nGrpWells = 4;
nDspWells = nGrpWells;
MSL_IniXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
MSL_SetAspStepVol( 0, 1, &M05030000_Aspirate, 0.0 );
MSL_SetDspStepVol( 0, 1, &M05030001_Dsp1_Dispende, MSL_CallFloatVarUf( "Uf_dDspVolume_S6",
&Rt_dDspVolume_S6, pPC ), 4 );
if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
{
    dXfrVol = MSL_CalAspStepVol( nDspWells );

    //-----
    // Aspirate
    //-----
    if( MSL_UseAspStep( 0, 1 ) )
    {
        dAspVol = MSL_GetAspStepVol( 0 );

        if( nDspWells == nGrpWells )
            dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
        else
            dWasteVol = 0;

        dDspVol = 0.0;

        nRet = EGS_AspEx( pPC, 88, &M05030000_Aspirate, 0,
            "B2_P", // sample id
            dAspVol, // aspirate volume
            0.0, // pre-aspirate air volume (for blowout)
            3.0, // post-aspirate air volume
            dDspVol, // dispense back volume
            dWasteVol ); // Waste volume

        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        if( nRet == -3 ) break; // Skip Sample? // Specify additional aspirate
        details...
        EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
        PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
        dLLSCLotDetHeight, 1 );
    }

    //-----
    // Dsp1:Dispense
    //-----
    if( MSL_UseDspStep( 0, 1, &nDspWells ) )
    {
        dDspVol = MSL_GetDspStepVol( 0 );

        dPostAir = 3.0;
        nDspMode = 0;

        nDspMax = 4; // Load maximum dispenses
        for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
        {
            nRet = EGS_DspEx( pPC, 89, &M05030001_Dsp1_Dispende, 1,
                dDspVol, // dispense volume
                dPostAir, // post air volume
                nDspMode, // dispense mode
                0 ); // Reserved for blowout delay

            if( nRet == -1 ) return nRet;
            if( nRet == -2 ) { bEOM = 1; break; }
            nDspWells--;
            // Specify additional dispense details...
            EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
            PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
        }
    }

    // Increment the transfer group count.
    nXfrGroups++;

    // Break out of transfer group of EOM reached.
    if( bEOM ) break;

```

```

    }

    // Clear the sequence if a skip sample was returned
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

    // Quit if EOM reached and no wells were dispensed
    if( bEOM && (nDspWells == nGrpWells) ) break;

    //-----
    // Flush_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 91, &M050301_Flush_1, 0,
                          dDspVol,      // system liquid volume
                          0.0,          // post-wash system air gap
                          1 );          // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // Wash_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 94, &M050302_Wash_1, 0,
                          dDspVol,      // system liquid volume
                          5.0,          // post-wash system air gap
                          1 );          // wash cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Backout the current sequence if no transfer groups were created.
    if( nXfrGroups == 0 )
    {
        EGS_ClearCurSeq( pPC );
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1;    // Abort the test.
    if( nRet == 2 ) return 0;     // Abort this procedure but not the test.
    pParentPC->nSeqCnt = pPC->nSeqCnt;

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

return 0;
} //<End P0503_Buffer__3_Panel()>

/*****
*
*   P0504_Buffer__4_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0504_Buffer__4_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0504_Buffer__4_Panel;
    MP2_PROC_CONTEXT_DEF *pParentPC = pParentPC;

```

```

int  nRet;                // Funct. return value
int  nFileIndex = pParentPC->nProcLoop; // Current File Index
int  nLoopCount = 0;      // Number of unskipped procedure loops
int  bDone;               // Last procedure loop flag
int  bEOM = 0;            // End of map indicator
int  nXfrGroups;          // Nbr of transfer groups constructed for the sample
int  nMask;               // General purpose mask
double dMixVol;           // Mix volume (µL)
double dAspVol;           // Aspirate volume (µL)
double dDspVol;           // Dispense volume (µL)
double dXfrVol;           // Total transfer volume (µL)
double dWasteVol = 0;     // Waste volume (µL)
double dBlowoutVol = 0;   // Blowout volume (µL)
double dPostAir;          // Post-step aspirate air gap (µL)
int  nDsp;                // Dispense counter
int  nDspMax;             // Maximum number of dispenses
int  nDspMode;            // Dispense mode
double dLLSAspHeight;     // Liq. Level Sense Aspirate Height
double dLLSDspHeight;     // Liq. Level Sense Dispense Height
double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
double dRetractFromLiqHeight; // Retract from liquid height
double dRetractFromLiqSpeed; // Retract from liquid speed
int  nGrpWells;           // Nbr of wells to dispense in xfr group
int  nDspWells;           // Remaining wells to dispense in xfr group

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );
EGS_SetLiquidType( "SyringeTest.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Start with caller specified sequence count.
pPC->nSeqCnt = pParentPC->nSeqCnt;

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 9, "B4", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );
}

```

```

// Start a dilutor sequence for this sample.
nRet = EGS_NewSequenceEx( pPC );
if( nRet == -1 ) return -1;
pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
if( pPC->hDilutor == -1 ) return -1;
pPC->nSeqCnt++;
pParentPC->nSeqCnt = pPC->nSeqCnt;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 2 ); // Waste Mode

//-----
// Transfer Group_1
//-----
nGrpWells = 4;
nDspWells = nGrpWells;
MSL_InitXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
MSL_SetAspStepVol( 0, 1, &M05040000_Aspirate, 0.0 );
MSL_SetDspStepVol( 0, 1, &M05040001_Dsp1_Dispende, MSL_CallFloatVarUf( "Uf_dDspVolume_S7",
&Rt_dDspVolume_S7, pPC ), 4 );
if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
{
    dXfrVol = MSL_CalAspStepVol( nDspWells );

    //-----
    // Aspirate
    //-----
    if( MSL_UseAspStep( 0, 1 ) )
    {
        dAspVol = MSL_GetAspStepVol( 0 );

        if( nDspWells == nGrpWells )
            dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
        else
            dWasteVol = 0;

        dDspVol = 0.0;

        nRet = EGS_AspEx( pPC, 100, &M05040000_Aspirate, 0,
            "B2_P", // sample id
            dAspVol, // aspirate volume
            0.0, // pre-aspirate air volume (for blowout)
            3.0, // post-aspirate air volume
            dDspVol, // dispense back volume
            dWasteVol ); // Waste volume

        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        if( nRet == -3 ) break; // Skip Sample? // Specify additional aspirate
details...
        EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
dLLSCLotDetHeight, 1 );
    }

    //-----
    // Dsp1:Dispense
    //-----
    if( MSL_UseDspStep( 0, 1, &nDspWells ) )
    {
        dDspVol = MSL_GetDspStepVol( 0 );

        dPostAir = 3.0;
        nDspMode = 0;

        nDspMax = 4; // Load maximum dispenses
        for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
        {
            nRet = EGS_DspEx( pPC, 101, &M05040001_Dsp1_Dispende, 1,
                dDspVol, // dispense volume
                dPostAir, // post air volume
                nDspMode, // dispense mode
                0 ); // Reserved for blowout delay
            if( nRet == -1 ) return nRet;

```

```

        if( nRet == -2 ) { bEOM = 1; break; }
        nDspWells--;
        // Specify additional dispense details...
        EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
    }
}

// Increment the transfer group count.

nXfrGroups++;

// Break out of transfer group of EOM reached.
if( bEOM ) break;
}

// Clear the sequence if a skip sample was returned
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

// Quit if EOM reached and no wells were dispensed
if( bEOM && (nDspWells == nGrpWells) ) break;

//-----
// Flush_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 103, &M050401_Flush_1, 0,
                        dDspVol,      // system liquid volume
                        0.0,          // post-wash system air gap
                        1 );          // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Wash_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 106, &M050402_Wash_1, 0,
                        dDspVol,      // system liquid volume
                        5.0,          // post-wash system air gap
                        1 );          // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Backout the current sequence if no transfer groups were created.
if( nXfrGroups == 0 )
{
    EGS_ClearCurSeq( pPC );
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;    // Abort the test.
if( nRet == 2 ) return 0;     // Abort this procedure but not the test.
pParentPC->nSeqCnt = pPC->nSeqCnt;

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

return 0;
} //<End P0504_Buffer__4_Panel()>

```

```

/*****
*
*   P05_Buffer_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P05_Buffer_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C05_Buffer_Panel;
    int nRet;           // Funct. return value
    int nFileIndex = 0;  // Current File Index
    int nLoopCount = 0;  // Number of unskipped procedure loops
    int bDone;           // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );

    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = 1;
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        //-----
        // Flush/Wash
        //-----
        nRet = MSL_CallChildProc( "P0500_Flush_Wash", nLoopCount, 0, 1, bDone );
        if( nRet == -1 ) return -1;

        //-----
        // Buffer #1 Panel
        //-----
        nRet = MSL_CallChildProc( "P0501_Buffer__1_Panel", nLoopCount, 0, 1, bDone );
        if( nRet == -1 ) return -1;

        //-----
        // Buffer #2 Panel
        //-----
        nRet = MSL_CallChildProc( "P0502_Buffer__2_Panel", nLoopCount, 0, 1, bDone );
        if( nRet == -1 ) return -1;

        //-----
        // Buffer #3 Panel
        //-----
        nRet = MSL_CallChildProc( "P0503_Buffer__3_Panel", nLoopCount, 0, 1, bDone );
        if( nRet == -1 ) return -1;

        //-----
        // Buffer #4 Panel
        //-----
        nRet = MSL_CallChildProc( "P0504_Buffer__4_Panel", nLoopCount, 0, 1, bDone );
        if( nRet == -1 ) return -1;

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );
    }
}

```

```

    } //<End Procedure Loop>

    // Execute any remaining sequences unconditionally.
    nRet = EGS_ExecuteEx( pPC, 1 );
    if( nRet == -1 ) return -1;    // Abort the test.

    return 0;

} //<End P05_Buffer_Panel()>

/*****
*
*   P0600_Flush_Wash_1()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0600_Flush_Wash_1()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0600_Flush_Wash_1;
    MP2_PROC_CONTEXT_DEF *pParentPC = pPC->pParentPC;
    int nRet;                // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;              // Last procedure loop flag
    int nMask;              // General purpose mask

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Start with caller specified sequence count.
    pPC->nSeqCnt = pParentPC->nSeqCnt;

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );

    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 1 );

        // Perform step 'Flush' for all dilutors
        nRet = MSL_WashAllDilutorsInProc(
            pPC, 113,
            &M060000_Flush_1,
            1, // Pump: 1=Peri, 0=Syringe
            3000.0, // System liquid volume
            0.0, // post-flush system air gap
            1, // wash cycles
            1, 1.0, 0.0, -1, 0, 10.0, -1, 0,
            "", // pre-step callback
            "" ); // post-step callback
        if( nRet == -1 ) return nRet;

        // Perform step 'Wash' for all dilutors
        nRet = MSL_WashAllDilutorsInProc(
            pPC, 116,

```

```

        &M060001_Wash_1,
        1,          // Pump: 1=Peri, 0=Syringe
        3000.0,      // System liquid volume
        5.0,        // post-flush system air gap
        1,          // wash cycles
        1, 1.0, 0.0, -1, 0, 10.0, -1, 0,
        "",         // pre-step callback
        "" );       // post-step callback
    if( nRet == -1 ) return nRet;

} //<End Procedure Loop>

return 0;

} //<End P0600_Flush_Wash_1()>

/*****
*
*   P0601_DHB_Matrix_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0601_DHB_Matrix_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0601_DHB_Matrix_Panel;
    MP2_PROC_CONTEXT_DEF *pParentPC = pPC->pParentPC;
    int nRet;          // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;         // Last procedure loop flag
    int bEOM = 0;      // End of map indicator
    int nXfrGroups;    // Nbr of transfer groups constructed for the sample
    int nMask;         // General purpose mask
    double dMixVol;     // Mix volume (µL)
    double dAspVol;     // Aspirate volume (µL)
    double dDspVol;     // Dispense volume (µL)
    double dXfrVol;     // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir;    // Post-step aspirate air gap (µL)
    int nDsp;           // Dispense counter
    int nDspMax;        // Maximum number of dispenses
    int nDspMode;       // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed
    int nGrpWells;      // Nbr of wells to dispense in xfr group
    int nDspWells;      // Remaining wells to dispense in xfr group

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );
    EGS_SetLiquidType( "SyringeTest.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Start with caller specified sequence count.
    pPC->nSeqCnt = pParentPC->nSeqCnt;

    // Load list of dilutor in use flags

```



```

strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 4, "DHB", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    pParentPC->nSeqCnt = pPC->nSeqCnt;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 2 ); // Waste Mode

    //-----
    // Transfer Group_1
    //-----
    nGrpWells = 4;
    nDspWells = nGrpWells;
    MSL_InitXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
    MSL_SetAspStepVol( 0, 1, &M06010000_Aspirate, 0.0 );
    MSL_SetDspStepVol( 0, 1, &M06010001_Dsp1_Dispense, MSL_CallFloatVarUf( "Uf_dDspVolume_S8",
&Rt_dDspVolume_S8, pPC ), 4 );
    if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
    {
        dXfrVol = MSL_CalAspStepVol( nDspWells );

        //-----
        // Aspirate
        //-----
        if( MSL_UseAspStep( 0, 1 ) )
        {
            dAspVol = MSL_GetAspStepVol( 0 );

            if( nDspWells == nGrpWells )
                dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
            else
                dWasteVol = 0;

            dDspVol = 0.0;

            nRet = EGS_AspEx( pPC, 122, &M06010000_Aspirate, 0,
                            "DHB_P", // sample id
                            dAspVol, // aspirate volume
                            0.0, // pre-aspirate air volume (for blowout)

```

```

        3.0,          // post-aspirate air volume
        dDspVol,      // dispense back volume
        dWasteVol );  // Waste volume
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) break;      // Skip Sample?           // Specify additional aspirate
details...
    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
    PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
    dLLSCLotDetHeight, 1 );
}

//-----
// Dsp1:Dispense
//-----
if( MSL_UseDspStep( 0, 1, &nDspWells ) )
{
    dDspVol = MSL_GetDspStepVol( 0 );

    dPostAir = 3.0;
    nDspMode = 0;

    nDspMax = 4;      // Load maximum dispenses
    for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
    {
        nRet = EGS_DspEx( pPC, 123, &M06010001_Dsp1_Dispense, 1,
            dDspVol,          // dispense volume
            dPostAir,         // post air volume
            nDspMode,         // dispense mode
            0 );              // Reserved for blowout delay
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        nDspWells--;
        // Specify additional dispense details...
        EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
        PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
    }

    // Increment the transfer group count.
    nXfrGroups++;

    // Break out of transfer group of EOM reached.
    if( bEOM ) break;
}

// Clear the sequence if a skip sample was returned
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

// Quit if EOM reached and no wells were dispensed
if( bEOM && (nDspWells == nGrpWells) ) break;

//-----
// Flush_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 125, &M060101_Flush_1, 0,
    dDspVol,          // system liquid volume
    0.0,             // post-wash system air gap
    1 );              // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Wash_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 128, &M060102_Wash_1, 0,
    dDspVol,          // system liquid volume
    5.0,             // post-wash system air gap
    1 );              // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...

```

```

EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Backout the current sequence if no transfer groups were created.
if( nXfrGroups == 0 )
{
    EGS_ClearCurSeq( pPC );
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1; // Abort the test.
if( nRet == 2 ) return 0; // Abort this procedure but not the test.
pParentPC->nSeqCnt = pPC->nSeqCnt;

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

return 0;

} //<End P0601_DHB_Matrix_Panel()>

/*****
*
*   P0602_alpha_Matrix_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0602_alpha_Matrix_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0602_alpha_Matrix_Panel;
    MP2_PROC_CONTEXT_DEF *pParentPC = pPC->pParentPC;
    int nRet; // Funct. return value
    int nFileIndex = pParentPC->nProcLoop; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed
    int nGrpWells; // Nbr of wells to dispense in xfr group
    int nDspWells; // Remaining wells to dispense in xfr group

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );

```

```

EGS_SetLiquidType( "SyringeTest.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Start with caller specified sequence count.
pPC->nSeqCnt = pParentPC->nSeqCnt;

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 5, "alpha", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    pParentPC->nSeqCnt = pPC->nSeqCnt;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 2 ); // Waste Mode

    //-----
    // Transfer Group_1
    //-----
    nGrpWells = 4;
    nDspWells = nGrpWells;
    MSL_IniXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
    MSL_SetAspStepVol( 0, 1, &M06020000_Aspirate, 0.0 );
    MSL_SetDspStepVol( 0, 1, &M06020001_Dspl_Dispense, MSL_CallFloatVarUf( "Uf_dDspVolume_S9",
&Rt_dDspVolume_S9, pPC ), 4 );
    if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
    {
        dXfrVol = MSL_CalAspStepVol( nDspWells );

        //-----
        // Aspirate
        //-----
        if( MSL_UseAspStep( 0, 1 ) )
        {
            dAspVol = MSL_GetAspStepVol( 0 );

```

```

    if( nDspWells == nGrpWells )
        dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
    else
        dWasteVol = 0;

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 134, &M06020000_Aspirate, 0,
                      "alpha_P", // sample id
                      dAspVol, // aspirate volume
                      0.0, // pre-aspirate air volume (for blowout)
                      3.0, // post-aspirate air volume
                      dDspVol, // dispense back volume
                      dWasteVol ); // Waste volume
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) break; // Skip Sample? // Specify additional aspirate
details...
    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
    PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
    dLLSclotDetHeight, 1 );
}

//-----
// Dsp1:Dispense
//-----
if( MSL_UseDspStep( 0, 1, &nDspWells ) )
{
    dDspVol = MSL_GetDspStepVol( 0 );

    dPostAir = 3.0;
    nDspMode = 0;

    nDspMax = 4; // Load maximum dispenses
    for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
    {
        nRet = EGS_DspEx( pPC, 135, &M06020001_Dsp1_Dispense, 1,
                          dDspVol, // dispense volume
                          dPostAir, // post air volume
                          nDspMode, // dispense mode
                          0 ); // Reserved for blowout delay
        if( nRet == -1 ) return nRet;

        if( nRet == -2 ) { bEOM = 1; break; }
        nDspWells--;
        // Specify additional dispense details...
        EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
        PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
    }

    // Increment the transfer group count.
    nXfrGroups++;

    // Break out of transfer group of EOM reached.
    if( bEOM ) break;
}

// Clear the sequence if a skip sample was returned
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

// Quit if EOM reached and no wells were dispensed
if( bEOM && (nDspWells == nGrpWells) ) break;

//-----
// Flush_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 137, &M060201_Flush_1, 0,
                      dDspVol, // system liquid volume
                      0.0, // post-wash system air gap
                      1 ); // flush cycles

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

```

```

//-----
// Wash_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 140, &M060202_Wash_1, 0,
                      dDspVol, // system liquid volume
                      5.0, // post-wash system air gap
                      1 ); // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Backout the current sequence if no transfer groups were created.
if( nXfrGroups == 0 )
{
    EGS_ClearCurSeq( pPC );
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1; // Abort the test.
if( nRet == 2 ) return 0; // Abort this procedure but not the test.
pParentPC->nSeqCnt = pPC->nSeqCnt;

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

return 0;

} //<End P0602_alpha_Matrix_Panel()>

/*****
*
* P06_Matrices_Panel()
*
* <Reserved for text entered into the procedure's comment page.>
*
*****/

int P06_Matrices_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C06_Matrices_Panel;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = 1;
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;
    }

```

```

// If at or past the last loop, set the done flag.
bDone = (pPC->nProcLoop >= pPC->nProcLimit);

//-----
// Flush/Wash_1
//-----
nRet = MSL_CallChildProc( "P0600_Flush_Wash_1", nLoopCount, 0, 1, bDone );
if( nRet == -1 ) return -1;

//-----
// DHB-Matrix Panel
//-----
nRet = MSL_CallChildProc( "P0601_DHB_Matrix_Panel", nLoopCount, 0, 1, bDone );
if( nRet == -1 ) return -1;

//-----
// alpha-Matrix Panel
//-----
nRet = MSL_CallChildProc( "P0602_alpha_Matrix_Panel", nLoopCount, 0, 1, bDone );
if( nRet == -1 ) return -1;

// If done, break now.
if( bDone ) break;

// Increment the loop count.
nLoopCount++;

// Reset the current procedure context.
MSL_SetCurrentProcContext( pPC, 1, 0 );

} //<End Procedure Loop>

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1; // Abort the test.

return 0;

} //<End P06_Matrices_Panel()>

/*****
*
*   P07_Organic_Wash_Matrices_and_Buffers()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P07_Organic_Wash_Matrices_and_Buffers()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C07_Organic_Wash_Matrices_and_Buffers;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSclotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed
    int nMaxWells; // Nbr of wells to dispense in reagent procedure
    int nOptWells; // Nbr of wells in the optimized dispense well map
    int nGrpWells; // Nbr of wells to dispense in xfr group

```

```

int  nDspWells;          // Remaining wells to dispense in xfr group

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "WaterWasteFT.prf" );
EGS_SetLiquidType( "WaterWasteFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Compute the max wells to dispense.
nMaxWells = 4;
pPC->nSamples = nMaxWells;

// Attempt to create an alternate dispense map.
nOptWells = MSL_CreateFilteredWellMap( pPC, &M070001_Dispense, nMaxWells, 1 );
if( nMaxWells < nOptWells ) nMaxWells = nOptWells;

// Reagent addition loop.
pPC->nProcLimit = nMaxWells;
for( pPC->nProcLoop = 0; nMaxWells > 0; pPC->nProcLoop = pPC->nProcLimit - nMaxWells )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 2 ); // Waste Mode

    //-----
    // Transfer Group_1
    //-----
    nGrpWells = 1;
    if( nGrpWells > nMaxWells ) nGrpWells = nMaxWells;
    nMaxWells -= nGrpWells;
    nDspWells = nGrpWells;
    MSL_IniXfrGroupEx( 1, 1, 1, nGrpWells, 0x60 ); // Setup XfrGroup info cache.
    MSL_SetAspStepVol( 0, 1, &M070000_Aspirate, 0.0 );

```



```

MSL_SetDspStepVol( 0, 1, &M070001_Dispense, 10.0, 1 );
if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
{
    dXfrVol = MSL_CalAspStepVol( nDspWells );

    //-----
    // Aspirate
    //-----
    if( MSL_UseAspStep( 0, 1 ) )
    {
        dAspVol = MSL_GetAspStepVol( 0 );

        dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;

        dDspVol = 0.0;

        nRet = EGS_AspEx( pPC, 147, &M070000_Aspirate, 0,
                        "", // sample id
                        dAspVol, // aspirate volume
                        0.0, // pre-aspirate air volume (for blowout)
                        3.0, // post-aspirate air volume
                        dDspVol, // dispense back volume
                        dWasteVol ); // Waste volume

        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        if( nRet == -3 ) break; // Skip Sample? // Specify additional aspirate
details...
        EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
        PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
        dLLSCLotDetHeight, 1 );
    }

    //-----
    // Dispense
    //-----
    if( MSL_UseDspStep( 0, 1, &nDspWells ) )
    {
        dDspVol = MSL_GetDspStepVol( 0 );

        nRet = EGS_DspEx( pPC, 150, &M070001_Dispense, 1,
                        dDspVol, // dispense volume
                        3.0, // post air volume
                        0, // dispense mode
                        0 ); // Reserved for blowout delay

        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        nDspWells--;
        // Specify additional dispense details...
        EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
        PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
    }

    //-----
    // Flush_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 153, &M070002_Flush_1, 0,
                        dDspVol, // system liquid volume
                        0.0, // post-wash system air gap
                        1 ); // flush cycles

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // Wash_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 156, &M070003_Wash_1, 0,
                        dDspVol, // system liquid volume
                        5.0, // post-wash system air gap
                        1 ); // wash cycles

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...

```

```

    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Increment the transfer group count.
    nXfrGroups++;

    // Break out of transfer group of EOM reached.
    if( bEOM ) break;
}

// Clear the sequence if a skip sample was returned
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

// Quit if EOM reached and no wells were dispensed
if( bEOM && (nDspWells == nGrpWells) ) break;

// Backout the current sequence if no transfer groups were created.
if( nXfrGroups == 0 )
{
    EGS_ClearCurSeq( pPC );
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;    // Abort the test.
if( nRet == 2 ) return 0;    // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );
} //<End Reagent Addition Loop>

// Reset the current procedure context one last time.
pPC->nProcLoop--;
MSL_SetCurrentProcContext( pPC, 1, 0 );
pPC->nProcLoop++;

// Delete the alternate dispense map.
MSL_DeleteAlternateWellMap( &M070001_Dispense );

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;
} //<End P07_Organic_Wash_Matrices_and_Buffers()>

/*****
*
*   P08_Reagent_Panel()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P08_Reagent_Panel()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C08_Reagent_Panel;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    int bEOM = 0;            // End of map indicator
    int nXfrGroups;          // Nbr of transfer groups constructed for the sample
    int nMask;               // General purpose mask
    double dMixVol;          // Mix volume (µL)
    double dAspVol;          // Aspirate volume (µL)
    double dDspVol;          // Dispense volume (µL)
    double dXfrVol;          // Total transfer volume (µL)

```

```

double dWasteVol = 0;    // Waste volume (µL)
double dBlowoutVol = 0; // Blowout volume (µL)
double dPostAir;        // Post-step aspirate air gap (µL)
int nDsp;               // Dispense counter
int nDspMax;            // Maximum number of dispenses
int nDspMode;           // Dispense mode
double dLLSAspHeight;   // Liq. Level Sense Aspirate Height
double dLLSDspHeight;   // Liq. Level Sense Dispense Height
double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
double dRetractFromLiqHeight; // Retract from liquid height
double dRetractFromLiqSpeed; // Retract from liquid speed
int nGrpWells;          // Nbr of wells to dispense in xfr group
int nDspWells;          // Remaining wells to dispense in xfr group

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SyringeTest.prf" );
EGS_SetLiquidType( "SyringeTest.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "N-N-N-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 10, "R1", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

```

```

// Load the pipette rule.
EGS_SetPipetteRule( 2 ); // Waste Mode

//-----
// Transfer Group_1
//-----
nGrpWells = 4;
nDspWells = nGrpWells;
MSL_IniXfrGroupEx( 1, 1, 1, nGrpWells, 0x0 ); // Setup XfrGroup info cache.
MSL_SetAspStepVol( 0, 1, &M080000_Aspirate, 0.0 );
MSL_SetDspStepVol( 0, 1, &M080001_Dsp1_Dispende, MSL_CallFloatVarUf( "Uf_dDspVolume_S10",
&Rt_dDspVolume_S10, pPC ), 4 );
if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
{
    dXfrVol = MSL_CalAspStepVol( nDspWells );

    //-----
    // Aspirate
    //-----
    if( MSL_UseAspStep( 0, 1 ) )
    {
        dAspVol = MSL_GetAspStepVol( 0 );

        if( nDspWells == nGrpWells )
            dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;
        else
            dWasteVol = 0;

        dDspVol = 0.0;

        nRet = EGS_AspEx( pPC, 163, &M080000_Aspirate, 0,
            "R1_P", // sample id
            dAspVol, // aspirate volume
            0.0, // pre-aspirate air volume (for blowout)
            3.0, // post-aspirate air volume
            dDspVol, // dispense back volume
            dWasteVol ); // Waste volume

        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        if( nRet == -3 ) break; // Skip Sample? // Specify additional aspirate
        details...
        EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
        PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
        dLLSCLotDetHeight, 1 );
    }

    //-----
    // Dsp1:Dispense
    //-----
    if( MSL_UseDspStep( 0, 1, &nDspWells ) )
    {
        dDspVol = MSL_GetDspStepVol( 0 );

        dPostAir = 3.0;
        nDspMode = 0;

        nDspMax = 4; // Load maximum dispenses
        for( nDsp = 0; (nDsp < nDspMax) && (nDspWells > 0); nDsp++ )
        {
            nRet = EGS_DspEx( pPC, 164, &M080001_Dsp1_Dispende, 1,
                dDspVol, // dispense volume
                dPostAir, // post air volume
                nDspMode, // dispense mode
                0 ); // Reserved for blowout delay

            if( nRet == -1 ) return nRet;
            if( nRet == -2 ) { bEOM = 1; break; }
            nDspWells--;
            // Specify additional dispense details...
            EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
            PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
        }
    }

    // Increment the transfer group count.
    nXfrGroups++;

    // Break out of transfer group of EOM reached.
    if( bEOM ) break;

```

```

    }

    // Clear the sequence if a skip sample was returned
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

    // Quit if EOM reached and no wells were dispensed
    if( bEOM && (nDspWells == nGrpWells) ) break;

    //-----
    // Flush_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 166, &M0801_Flush_1, 0,
                          dDspVol, // system liquid volume
                          0.0, // post-wash system air gap
                          1 ); // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // Wash_1
    //-----
    dDspVol = 3000.0;
    nRet = EGS_PeriWashEx( pPC, 169, &M0802_Wash_1, 0,
                          dDspVol, // system liquid volume
                          5.0, // post-wash system air gap
                          1 ); // wash cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Backout the current sequence if no transfer groups were created.
    if( nXfrGroups == 0 )
    {
        EGS_ClearCurSeq( pPC );
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1; // Abort the test.
    if( nRet == 2 ) return 0; // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...

    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1; // Abort the test.

return 0;

} //<End P08_Reagent_Panel()>

/*****
*
* P09_Organic_Wash_Reagent()
*
* <Reserved for text entered into the procedure's comment page.>
*
*****/

int P09_Organic_Wash_Reagent()
```

```

{
    MP2_PROC_CONTEXT_DEF *pPC = &C09_Organic_Wash_Reagent;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed
    int nMaxWells; // Nbr of wells to dispense in reagent procedure
    int nOptWells; // Nbr of wells in the optimized dispense well map
    int nGrpWells; // Nbr of wells to dispense in xfr group
    int nDspWells; // Remaining wells to dispense in xfr group

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "WaterWasteFT.prf" );
    EGS_SetLiquidType( "WaterWasteFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "N-N-N-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Compute the max wells to dispense.
    nMaxWells = 1;
    pPC->nSamples = nMaxWells;

    // Attempt to create an alternate dispense map.
    nOptWells = MSL_CreateFilteredWellMap( pPC, &M090001_Dispense, nMaxWells, 1 );
    if( nMaxWells < nOptWells ) nMaxWells = nOptWells;

    // Reagent addition loop.
    pPC->nProcLimit = nMaxWells;
    for( pPC->nProcLoop = 0; nMaxWells > 0; pPC->nProcLoop = pPC->nProcLimit - nMaxWells )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 8, "R1", 1, NULL ) )

```

```

{
    MSL_IncProcSampleMaps( pPC );
    nMaxWells -= 1;
    continue;
}

// Increment the loop count.
nLoopCount++;

// Reset the current procedure context.
MSL_SetCurrentProcContext( pPC, 1, 1 );

// Start a dilutor sequence for this sample.
nRet = EGS_NewSequenceEx( pPC );
if( nRet == -1 ) return -1;
pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
if( pPC->hDilutor == -1 ) return -1;
pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 2 ); // Waste Mode

//-----
// Transfer Group_1
//-----
nGrpWells = 1;
if( nGrpWells > nMaxWells ) nGrpWells = nMaxWells;
nMaxWells -= nGrpWells;
nDspWells = nGrpWells;
MSL_IniXfrGroupEx( 1, 1, 1, nGrpWells, 0x60 ); // Setup XfrGroup info cache.
MSL_SetAspStepVol( 0, 1, &M090000_Aspirate, 0.0 );
MSL_SetDspStepVol( 0, 1, &M090001_Dispense, 10.0, 1 );
if( MSL_UseXfrGroup( &bEOM, 1, &nDspWells ) ) while( nDspWells > 0 )
{
    dXfrVol = MSL_CalAspStepVol( nDspWells );

    //-----
    // Aspirate
    //-----
    if( MSL_UseAspStep( 0, 1 ) )
    {
        dAspVol = MSL_GetAspStepVol( 0 );

        dWasteVol = PrfFileGetWastePercent( pPC->hPrfFile, dXfrVol ) * dXfrVol / 100.0;

        dDspVol = 0.0;

        nRet = EGS_AspEx( pPC, 175, &M090000_Aspirate, 0,
                        "", // sample id
                        dAspVol, // aspirate volume
                        0.0, // pre-aspirate air volume (for blowout)
                        3.0, // post-aspirate air volume
                        dDspVol, // dispense back volume
                        dWasteVol ); // Waste volume

        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        if( nRet == -3 ) break; // Skip Sample? // Specify additional aspirate
details...
        EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
        PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0,
        dLLSCLotDetHeight, 1 );
    }

    //-----
    // Dispense
    //-----
    if( MSL_UseDspStep( 0, 1, &nDspWells ) )
    {
        dDspVol = MSL_GetDspStepVol( 0 );

        nRet = EGS_DspEx( pPC, 178, &M090001_Dispense, 1,
                        dDspVol, // dispense volume
                        3.0, // post air volume

```

```

        0,          // dispense mode
        0 );      // Reserved for blowout delay
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    nDspWells--;
    // Specify additional dispense details...
    EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ),
PrfFileGetDspDelay( pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );
}

//-----
// Flush_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 181, &M090002_Flush_1, 0,
                        dDspVol,          // system liquid volume
                        0.0,              // post-wash system air gap
                        1 );              // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Wash_1
//-----
dDspVol = 3000.0;
nRet = EGS_PeriWashEx( pPC, 184, &M090003_Wash_1, 0,
                        dDspVol,          // system liquid volume
                        5.0,              // post-wash system air gap
                        1 );              // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Increment the transfer group count.
nXfrGroups++;

// Break out of transfer group of EOM reached.
if( bEOM ) break;
}

// Clear the sequence if a skip sample was returned
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }

// Quit if EOM reached and no wells were dispensed
if( bEOM && (nDspWells == nGrpWells) ) break;

// Backout the current sequence if no transfer groups were created.
if( nXfrGroups == 0 )
{
    EGS_ClearCurSeq( pPC );
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;    // Abort the test.
if( nRet == 2 ) return 0;     // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Reagent Addition Loop>

// Reset the current procedure context one last time.
pPC->nProcLoop--;
MSL_SetCurrentProcContext( pPC, 1, 0 );
pPC->nProcLoop++;

// Delete the alternate dispense map.
MSL_DeleteAlternateWellMap( &M090001_Dispense );

// Backout the current sequence if an <End of Map> was reached.

```



```

    if( bEOM && (pPC->nSeqCnt > 0) )
        EGS_ClearCurSeq( pPC );

    // Execute any remaining sequences unconditionally.
    nRet = EGS_ExecuteEx( pPC, 1 );
    if( nRet == -1 ) return -1;    // Abort the test.

    return 0;
} //<End P09_Organic_Wash_Reagent()>

/*****
*
*   P0A_Aqueous_Flush_Wash()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0A_Aqueous_Flush_Wash()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0A_Aqueous_Flush_Wash;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    int nMask;               // General purpose mask

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 1 );

        // Perform step 'Flush' for all dilutors
        nRet = MSL_WashAllDilutorsInProc(
            pPC, 190,
            &M0A00_Flush_1,
            1,          // Pump: 1=Peri, 0=Syringe
            3000.0,      // System liquid volume
            0.0,         // post-flush system air gap
            1,          // wash cycles
            1, 1.0, 0.0, -1, 0, 10.0, -1, 0,
            "",          // pre-step callback
            "" );        // post-step callback
        if( nRet == -1 ) return nRet;

        // Perform step 'Wash' for all dilutors
        nRet = MSL_WashAllDilutorsInProc(
            pPC, 193,
            &M0A01_Wash_1,

```

```

        1,          // Pump: 1=Peri, 0=Syringe
        3000.0,      // System liquid volume
        5.0,        // post-flush system air gap
        1,          // wash cycles
        1, 1.0, 0.0, -1, 0, 10.0, -1, 0,
        "",         // pre-step callback
        "" );       // post-step callback
    if( nRet == -1 ) return nRet;

} //<End Procedure Loop>

return 0;

} //<End POA_Aqueous_Flush_Wash()>

/*****
*
*   POB_Sample_Spotting_Time_Marker()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int POB_Sample_Spotting_Time_Marker()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0B_Sample_Spotting_Time_Marker;
    int nRet;          // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;         // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Create/Update the Time Marker
        MSL_TimeMarkerCreate( pPC->szName );

    } //<End Procedure Loop>

    return 0;

} //<End POB_Sample_Spotting_Time_Marker()>

/*****
*
*   POC_Samples_to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

```

```

*****/

int POC_Samples_to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0C_Samples_to_MALDI26;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );

    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 1 );

        // Start a dilutor sequence for this sample.
        nRet = EGS_NewSequenceEx( pPC );
        if( nRet == -1 ) return -1;
    }
}

```



```

        1 );      // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 208, &M0C04_Aspirate_OrganicWash_LVT, 0,
                "OW",      // sample id
                dAspVol,    // aspirate volume
                dBlowoutVol, // pre-aspirate air volume (for blowout)
                3.0,        // post-aspirate air volume
                dDspVol,    // dispense back volume
                0.0 );      // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

//-----
// Dispense OrganicWash-LVT
//-----
dDspVol = 10.0;

nRet = EGS_DspEx( pPC, 211, &M0C05_Dispense_OrganicWash_LVT, 1,
                dDspVol,    // dispense volume
                3.0,        // post air volume
                1,          // dispense mode
                PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 214, &M0C06_AqueousFlush_after_OrganicWash_LVT, 0,
                    dDspVol,    // system liquid volume
                    5.0,        // post-wash system air gap
                    1 );        // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 217, &M0C07_AqueousWash_after_OrganicWash_LVT, 0,
                    dDspVol,    // system liquid volume
                    5.0,        // post-wash system air gap
                    1 );        // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

```

```

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1;    // Abort the test.
    if( nRet == 2 ) return 0;    // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

} //<End POD_Samples_to_MALDI26()>

/*****
*
*   POD_Calibrant_Time_Marker()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int POD_Calibrant_Time_Marker()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0D_Calibrant_Time_Marker;
    int nRet;           // Funct. return value
    int nFileIndex = 0;  // Current File Index
    int nLoopCount = 0;  // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.

    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 15, "C1", 1, NULL ) )
        {
            MSL_IncProcSampleMaps( pPC );
            continue;
        }

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Create/Update the Time Marker
        MSL_TimeMarkerCreate( pPC->szName );
    }
}

```

```

    } //<End Procedure Loop>

    return 0;

} //<End POD_Calibrant_Time_Marker()>

/*****
*
*   POE_Calibrant_to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int POE_Calibrant_to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0E_Calibrant_to_MALDI26;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)

    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSclotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSclotDetHeight = PrfFileGetLLSclotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "N-Y-N-N" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = 1;
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);
    }

```

```

// Child procedure calls would be written here...

// If done, break now.
if( bDone ) break;

// Should this loop be skipped?
if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 17, "C1", 1, NULL ) )
{
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Increment the loop count.
nLoopCount++;

// Reset the current procedure context.
MSL_SetCurrentProcContext( pPC, 1, 1 );

// Start a dilutor sequence for this sample.
nRet = EGS_NewSequenceEx( pPC );
if( nRet == -1 ) return -1;
pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
if( pPC->hDilutor == -1 ) return -1;
pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate Calibrand
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 3, 1.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 224, &M0E00_Aspirate_Calibrand, 0,
                  "Call", // sample id
                  dAspVol, // aspirate volume
                  dBlowoutVol, // pre-aspirate air volume (for blowout)
                  3.0, // post-aspirate air volume
                  dDspVol, // dispense back volume
                  0.0 ); // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Calibrand
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 3, 1.0 );

nRet = EGS_DspEx( pPC, 225, &M0E01_Dispense_Calibrand, 1,
                  dDspVol, // dispense volume
                  3.0, // post air volume
                  1, // dispense mode
                  PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----

```



```

// AqueousFlush before OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 226, &M0E02_AqueousFlush_before_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // flush cycles

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash before OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 229, &M0E03_AqueousWash_before_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 232, &M0E04_Aspirate_OrganicWash_LVT, 0,
                 "OW",      // sample id
                 dAspVol,    // aspirate volume
                 dBlowoutVol, // pre-aspirate air volume (for blowout)
                 3.0,        // post-aspirate air volume
                 dDspVol,    // dispense back volume
                 0.0 );      // No waste

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

//-----
// Dispense OrganicWash-LVT
//-----
dDspVol = 10.0;

nRet = EGS_DspEx( pPC, 235, &M0E05_Dispense_OrganicWash_LVT, 1,
                 dDspVol,      // dispense volume
                 3.0,          // post air volume
                 1,            // dispense mode
                 PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 238, &M0E06_AqueousFlush_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // flush cycles

```

```

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // AqueousWash after OrganicWash-LVT
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 241, &M0E07_AqueousWash_after_OrganicWash_LVT, 0,
                          dDspVol, // system liquid volume
                          5.0, // post-wash system air gap
                          1 ); // wash cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1; // Abort the test.
    if( nRet == 2 ) return 0; // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1; // Abort the test.

return 0;

} //<End P0E_Calibrant_to_MALDI26()>

/*****
*
* P0F_Sample_Drying_Timer()
*
* <Reserved for text entered into the procedure's comment page.>
*
*****/

int P0F_Sample_Drying_Timer()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C0F_Sample_Drying_Timer;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.

```

```

        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Sample Spotting Time Marker", MSL_CallTimeVarUf( "Uf_stTimePeriod_S1",
&Rt_stTimePeriod_S1, pPC ), NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;
} //<End P0F_Sample_Drying_Timer()>

/*****
*
*   P10_Buffer__1_Spotting_Time_Marker()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P10_Buffer__1_Spotting_Time_Marker()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C10_Buffer__1_Spotting_Time_Marker;
    int  nRet;           // Funct. return value
    int  nFileIndex = 0; // Current File Index
    int  nLoopCount = 0; // Number of unskipped procedure loops
    int  bDone;          // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Create/Update the Time Marker
        MSL_TimeMarkerCreate( pPC->szName );

    } //<End Procedure Loop>

    return 0;
} //<End P10_Buffer__1_Spotting_Time_Marker()>

/*****
*
*   P11_Buffer__1_to_MALDI26()

```

```

*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P11_Buffer__1_to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C11_Buffer__1_to_MALDI26;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 5, "B1", 1, NULL ) )
        {
            MSL_IncProcSampleMaps( pPC );
            continue;
        }
    }
}

```

```

// Increment the loop count.
nLoopCount++;

// Reset the current procedure context.
MSL_SetCurrentProcContext( pPC, 1, 1 );

// Start a dilutor sequence for this sample.
nRet = EGS_NewSequenceEx( pPC );
if( nRet == -1 ) return -1;
pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
if( pPC->hDilutor == -1 ) return -1;
pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate Buffer #1
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 4, 10.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 250, &M1100_Aspirate_Buffer__1, 1,
id      MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume
        dDspVol, // dispense back volume
        0.0 ); // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Buffer #1
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 4, 10.0 );

nRet = EGS_DspEx( pPC, 251, &M1101_Dispense_Buffer__1, 0,
        dDspVol, // dispense volume
        3.0, // post air volume
        1, // dispense mode
        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush before OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 252, &M1102_AqueousFlush_before_OrganicWash_LVT, 0,
        dDspVol, // system liquid volume
        5.0, // post-wash system air gap
        1 ); // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

```

```

//-----
// AqueousWash before OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 255, &M1103_AqueousWash_before_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 258, &M1104_Aspirate_OrganicWash_LVT, 0,
                 "OW",      // sample id
                 dAspVol,    // aspirate volume
                 dBlowoutVol, // pre-aspirate air volume (for blowout)
                 3.0,        // post-aspirate air volume
                 dDspVol,    // dispense back volume
                 0.0 );      // No waste

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

//-----
// Dispense OrganicWash-LVT
//-----
dDspVol = 10.0;

nRet = EGS_DspEx( pPC, 261, &M1105_Dispense_OrganicWash_LVT, 1,
                 dDspVol,      // dispense volume
                 3.0,          // post air volume
                 1,            // dispense mode
                 PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay

if( nRet == -1 ) return nRet;

if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 264, &M1106_AqueousFlush_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // flush cycles

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 267, &M1107_AqueousWash_after_OrganicWash_LVT, 0,

```

```

        dDspVol,      // system liquid volume
        5.0,         // post-wash system air gap
        1 );        // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;    // Abort the test.
if( nRet == 2 ) return 0;     // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

} //<End P11_Buffer__1_to_MALDI26()>

/*****
*
*   P12_Buffer__1_Timer()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P12_Buffer__1_Timer()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C12_Buffer__1_Timer;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    MP2_TIME stTime;    // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.

```

```

        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Buffer #1 Spotting Time Marker", MSL_GetRtFileColTime( &RtFile3,
nFileIndex, 11, &stTime ), NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;

} //<End P12_Buffer__1_Timer()>

/*****
*
*   P13_Reagent_Spotting_Time_Marker()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P13_Reagent_Spotting_Time_Marker()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C13_Reagent_Spotting_Time_Marker;
    int nRet;           // Funct. return value
    int nFileIndex = 0;  // Current File Index
    int nLoopCount = 0;  // Number of unskipped procedure loops
    int bDone;           // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Create/Update the Time Marker
        MSL_TimeMarkerCreate( pPC->szName );

    } //<End Procedure Loop>

    return 0;

} //<End P13_Reagent_Spotting_Time_Marker()>

/*****
*
*   P14_Reagent_to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P14_Reagent_to_MALDI26()

```



```

{
    MP2_PROC_CONTEXT_DEF *pPC = &C14_Reagent_to_MALDI26;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 9, "R1", 1, NULL ) )
        {
            MSL_IncProcSampleMaps( pPC );
            continue;
        }

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.

```

```

MSL_SetCurrentProcContext( pPC, 1, 1 );

// Start a dilutor sequence for this sample.
nRet = EGS_NewSequenceEx( pPC );
if( nRet == -1 ) return -1;
pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
if( pPC->hDilutor == -1 ) return -1;
pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate Reagent
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 8, 10.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 276, &M1400_Aspirate_Reagent, 1,
id      MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume
        dDspVol, // dispense back volume
        0.0 ); // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Reagent
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 8, 10.0 );

nRet = EGS_DspEx( pPC, 277, &M1401_Dispende_Reagent, 0,
        dDspVol, // dispense volume
        3.0, // post air volume
        1, // dispense mode
        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush before OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 278, &M1402_AqueousFlush_before_OrganicWash_LVT, 0,
        dDspVol, // system liquid volume
        5.0, // post-wash system air gap
        1 ); // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash before OrganicWash-LVT
//-----

```

```

dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 281, &M1403_AqueousWash_before_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 284, &M1404_Aspirate_OrganicWash_LVT, 0,
                 "OW",      // sample id
                 dAspVol,    // aspirate volume
                 dBlowoutVol, // pre-aspirate air volume (for blowout)
                 3.0,        // post-aspirate air volume
                 dDspVol,    // dispense back volume
                 0.0 );      // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0, 0, 0, dLLSCLotDetHeight, 1 );

//-----
// Dispense OrganicWash-LVT
//-----
dDspVol = 10.0;

nRet = EGS_DspEx( pPC, 287, &M1405_Dispense_OrganicWash_LVT, 1,
                 dDspVol,      // dispense volume
                 3.0,          // post air volume
                 1,            // dispense mode
                 PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 290, &M1406_AqueousFlush_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 293, &M1407_AqueousWash_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

```

```

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;    // Abort the test.
if( nRet == 2 ) return 0;    // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

} //<End P14_Reagent_to_MALDI26()>

/*****
*
*   P15_Reaction_Timer()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P15_Reaction_Timer()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C15_Reaction_Timer;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    MP2_TIME stTime;         // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Reagent Spotting Time Marker", MSL_GetRtFileColTime( &RtFile3,
nFileIndex, 13, &stTime ), NULL, 0 );

```

```

        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;

} //<End P15_Reaction_Timer()>

/*****
*
*   P16_Buffer__2_Spotting_Time_Marker()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P16_Buffer__2_Spotting_Time_Marker()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C16_Buffer__2_Spotting_Time_Marker;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Create/Update the Time Marker
        MSL_TimeMarkerCreate( pPC->szName );

    } //<End Procedure Loop>

    return 0;

} //<End P16_Buffer__2_Spotting_Time_Marker()>

/*****
*
*   P17_Buffer__2_to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P17_Buffer__2_to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C17_Buffer__2_to_MALDI26;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops

```

```

int  bDone;           // Last procedure loop flag
int  bEOM             = 0; // End of map indicator
int  nXfrGroups;      // Nbr of transfer groups constructed for the sample
int  nMask;           // General purpose mask
double dMixVol;       // Mix volume (µL)
double dAspVol;       // Aspirate volume (µL)
double dDspVol;       // Dispense volume (µL)
double dXfrVol;       // Total transfer volume (µL)
double dWasteVol = 0; // Waste volume (µL)
double dBlowoutVol = 0; // Blowout volume (µL)
double dPostAir;      // Post-step aspirate air gap (µL)
int  nDsp;            // Dispense counter

int  nDspMax;         // Maximum number of dispenses
int  nDspMode;        // Dispense mode
double dLLSAspHeight; // Liq. Level Sense Aspirate Height
double dLLSDspHeight; // Liq. Level Sense Dispense Height
double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
double dRetractFromLiqHeight; // Retract from liquid height
double dRetractFromLiqSpeed; // Retract from liquid speed

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 7, "B2", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
}

```

```

    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate Buffer #2
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 6, 10.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 302, &M1700_Aspirate_Buffer_2, 1,
        MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
id
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume
        dDspVol, // dispense back volume
        0.0 ); // No waste
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed
    if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
    // Specify additional aspirate details...
    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
    PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
    dLLSCLotDetHeight, 1 );

    //-----
    // Dispense Buffer #2
    //-----
    dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 6, 10.0 );

    nRet = EGS_DspEx( pPC, 303, &M1701_Dispende_Buffer_2, 0,
        dDspVol, // dispense volume
        3.0, // post air volume
        1, // dispense mode
        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    // Specify additional dispense details...
    EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

    //-----
    // AqueousFlush before OrganicWash-LVT
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 304, &M1702_AqueousFlush_before_OrganicWash_LVT, 0,
        dDspVol, // system liquid volume
        5.0, // post-wash system air gap
        1 ); // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // AqueousWash before OrganicWash-LVT
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 307, &M1703_AqueousWash_before_OrganicWash_LVT, 0,
        dDspVol, // system liquid volume
        5.0, // post-wash system air gap
        1 ); // wash cycles
    if( nRet == -1 ) return nRet;

```

```

if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 310, &M1704_Aspirate_OrganicWash_LVT, 0,
                  "OW",      // sample id
                  dAspVol,    // aspirate volume
                  dBlowoutVol, // pre-aspirate air volume (for blowout)
                  3.0,        // post-aspirate air volume
                  dDspVol,    // dispense back volume
                  0.0 );      // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetDspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

//-----
// Dispense OrganicWash-LVT
//-----
dDspVol = 10.0;

nRet = EGS_DspEx( pPC, 313, &M1705_Dispense_OrganicWash_LVT, 1,
                  dDspVol,    // dispense volume
                  3.0,        // post air volume
                  1,          // dispense mode
                  PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 316, &M1706_AqueousFlush_after_OrganicWash_LVT, 0,
                       dDspVol,    // system liquid volume
                       5.0,        // post-wash system air gap
                       1 );        // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 319, &M1707_AqueousWash_after_OrganicWash_LVT, 0,
                       dDspVol,    // system liquid volume
                       5.0,        // post-wash system air gap
                       1 );        // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.

```



```

    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1;    // Abort the test.
    if( nRet == 2 ) return 0;    // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

} //<End P17_Buffer__2_to_MALDI26()>

/*****
*
*   P18_Buffer__2_Timer()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P18_Buffer__2_Timer()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C18_Buffer__2_Timer;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    MP2_TIME stTime;         // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Buffer #2 Spotting Time Marker", MSL_GetRtFileColTime( &RtFile3,
nFileIndex, 12, &stTime ), NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;

```

```

} //<End P18_Buffer__2_Timer()>

/*****
*
*   P19_Enzyme_Pre_Incubation_Time()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P19_Enzyme_Pre_Incubation_Time()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C19_Enzyme_Pre_Incubation_Time;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    MP2_TIME stTime;    // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Enzyme Pre-Incubation", MSL_GetRtFileColTime( &RtFile3, nFileIndex,
19, &stTime ), NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;
} //<End P19_Enzyme_Pre_Incubation_Time()>

/*****
*
*   P1A_Enzyme_1_RTM()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P1A_Enzyme_1_RTM()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C1A_Enzyme_1_RTM;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index

```

```

int  nLoopCount = 0;    // Number of unskipped procedure loops
int  bDone;           // Last procedure loop flag

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Procedure loop.
pPC->nProcLimit = 1;
for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 0 );

    // Create/Update the Time Marker
    MSL_TimeMarkerCreate( pPC->szName );

} //<End Procedure Loop>

return 0;

} //<End P1A_Enzyme_1_RTM()>

/*****
*
*   P1B_Enzyme__1_to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P1B_Enzyme__1_to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C1B_Enzyme__1_to_MALDI26;
    int  nRet;           // Funct. return value
    int  nFileIndex = 0; // Current File Index
    int  nLoopCount = 0; // Number of unskipped procedure loops
    int  bDone;         // Last procedure loop flag
    int  bEOM           = 0; // End of map indicator
    int  nXfrGroups;    // Nbr of transfer groups constructed for the sample
    int  nMask;         // General purpose mask
    double dMixVol;     // Mix volume (µL)
    double dAspVol;     // Aspirate volume (µL)
    double dDspVol;     // Dispense volume (µL)
    double dXfrVol;     // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir;    // Post-step aspirate air gap (µL)
    int  nDsp;          // Dispense counter
    int  nDspMax;       // Maximum number of dispenses
    int  nDspMode;      // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

```

```

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 11, "E1", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate Enzyme #1
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 10, 1.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 330, &M1B00_Aspirate_Enzyme__1, 1,
        MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)

```

id

```

        3.0,      // post-aspirate air volume
        dDspVol, // dispense back volume
        0.0 );   // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Enzyme #1
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 10, 1.0 );

nRet = EGS_DspEx( pPC, 331, &M1B01_Dispense_Enzyme__1, 0,
        dDspVol, // dispense volume
        3.0,     // post air volume
        1,       // dispense mode
        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 332, &M1B02_AqueousFlush, 0,
        dDspVol, // system liquid volume
        5.0,     // post-wash system air gap
        1 );     // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 335, &M1B03_AqueousWash, 0,
        dDspVol, // system liquid volume
        5.0,     // post-wash system air gap
        1 );     // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1; // Abort the test.
if( nRet == 2 ) return 0;   // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1; // Abort the test.

```

```

    return 0;

} //<End PlB_Enzyme__1_to_MALDI26()>

/*****
*
*   PlC_Enzyme_1_IT()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int PlC_Enzyme_1_IT()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C1C_Enzyme_1_IT;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    MP2_TIME stTime;         // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;
    MSL_StringToTime( "0 0:03:00", &stTime );

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Enzyme_1_RTM", &stTime, NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;

} //<End PlC_Enzyme_1_IT()>

/*****
*
*   PlD_Buffer__3__E1__to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int PlD_Buffer__3__E1__to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C1D_Buffer__3__E1__to_MALDI26;
    int nRet;                // Funct. return value

```

```

int  nFileIndex = 0;      // Current File Index
int  nLoopCount = 0;      // Number of unskipped procedure loops
int  bDone;              // Last procedure loop flag
int  bEOM = 0;           // End of map indicator
int  nXfrGroups;          // Nbr of transfer groups constructed for the sample
int  nMask;               // General purpose mask
double dMixVol;           // Mix volume (µL)
double dAspVol;           // Aspirate volume (µL)
double dDspVol;           // Dispense volume (µL)
double dXfrVol;           // Total transfer volume (µL)
double dWasteVol = 0;     // Waste volume (µL)
double dBlowoutVol = 0;   // Blowout volume (µL)
double dPostAir;          // Post-step aspirate air gap (µL)
int  nDsp;                // Dispense counter

int  nDspMax;             // Maximum number of dispenses
int  nDspMode;            // Dispense mode
double dLLSAspHeight;     // Liq. Level Sense Aspirate Height
double dLLSDspHeight;     // Liq. Level Sense Dispense Height
double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
double dRetractFromLiqHeight; // Retract from liquid height
double dRetractFromLiqSpeed; // Retract from liquid speed

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 22, "B3", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );

```

```

if( nRet == -1 ) return -1;
pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
if( pPC->hDilutor == -1 ) return -1;
pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate Buffer #2
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 21, 10.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 342, &M1D00_Aspirate_Buffer_2, 1,
    MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
    dAspVol, // aspirate volume
    dBlowoutVol, // pre-aspirate air volume (for blowout)
    3.0, // post-aspirate air volume
    dDspVol, // dispense back volume
    0.0 ); // No waste

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSclotDetHeight, 1 );

//-----
// Dispense Buffer #2
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 21, 10.0 );

nRet = EGS_DspEx( pPC, 343, &M1D01_Dispense_Buffer_2, 0,
    dDspVol, // dispense volume
    3.0, // post air volume
    1, // dispense mode
    PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 344, &M1D02_AqueousFlush, 0,
    dDspVol, // system liquid volume
    5.0, // post-wash system air gap
    1 ); // flush cycles

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 347, &M1D03_AqueousWash, 0,
    dDspVol, // system liquid volume
    5.0, // post-wash system air gap

```



```

        1 );      // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;      // Abort the test.
if( nRet == 2 ) return 0;      // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;      // Abort the test.

return 0;

} //<End P1D_Buffer__3__E1__to_MALDI26()>

/*****
*
*   P1E_Buffer__4__E1__to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P1E_Buffer__4__E1__to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C1E_Buffer__4__E1__to_MALDI26;
    int nRet;          // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    int bEOM = 0;       // End of map indicator
    int nXfrGroups;     // Nbr of transfer groups constructed for the sample
    int nMask;          // General purpose mask
    double dMixVol;      // Mix volume (µL)
    double dAspVol;      // Aspirate volume (µL)
    double dDspVol;      // Dispense volume (µL)
    double dXfrVol;      // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir;     // Post-step aspirate air gap (µL)
    int nDsp;            // Dispense counter
    int nDspMax;         // Maximum number of dispenses
    int nDspMode;        // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );

```

```

dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSclotDetHeight  = PrfFileGetLLSclotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 24, "B4", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate Buffer #2
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 23, 10.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 352, &M1E00_Aspirate_Buffer__2, 1,
        MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
id
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume
        dDspVol, // dispense back volume
        0.0 ); // No waste
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed

```

```

        if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
        // Specify additional aspirate details...
        EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
        PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
        dLLSCLotDetHeight, 1 );

        //-----
        // Dispense Buffer #2
        //-----
        dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 23, 10.0 );

        nRet = EGS_DspEx( pPC, 353, &M1E01_Dispense_Buffer__2, 0,
                        dDspVol, // dispense volume
                        3.0, // post air volume
                        1, // dispense mode
                        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        // Specify additional dispense details...
        EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
        pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

        //-----
        // AqueousFlush
        //-----
        dDspVol = 2000.0;
        nRet = EGS_PeriWashEx( pPC, 354, &M1E02_AqueousFlush, 0,
                        dDspVol, // system liquid volume
                        5.0, // post-wash system air gap
                        1 ); // flush cycles
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }

        // Specify additional dispense details...
        EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

        //-----
        // AqueousWash
        //-----
        dDspVol = 2000.0;
        nRet = EGS_PeriWashEx( pPC, 357, &M1E03_AqueousWash, 0,
                        dDspVol, // system liquid volume
                        5.0, // post-wash system air gap
                        1 ); // wash cycles
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }

        // Specify additional dispense details...
        EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

        // Execute the accumulated sequences only if necessary.
        nRet = EGS_ExecuteEx( pPC, 0 );
        if( nRet == -1 ) return -1; // Abort the test.
        if( nRet == 2 ) return 0; // Abort this procedure but not the test.

        // Increment all per-sample maps here (if any)...
        MSL_IncProcSampleMaps( pPC );

    } //<End Procedure Loop>

    // Backout the current sequence if an <End of Map> was reached.
    if( bEOM && (pPC->nSeqCnt > 0) )
        EGS_ClearCurSeq( pPC );

    // Execute any remaining sequences unconditionally.
    nRet = EGS_ExecuteEx( pPC, 1 );
    if( nRet == -1 ) return -1; // Abort the test.

    return 0;
} //<End PlE_Buffer__4__E1__to_MALDI26()>

/*****
*
* PlF_Enzyme_1_RT()
*
*****/

```

```

*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P1F_Enzyme_1_RT()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C1F_Enzyme_1_RT;
    int nRet;           // Funct. return value
    int nFileIndex = 0;  // Current File Index
    int nLoopCount = 0;  // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    MP2_TIME stTime;     // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Enzyme_1_RTM", MSL_GetRtFileColTime( &RtFile3, nFileIndex, 14, &stTime
), NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;
} //<End P1F_Enzyme_1_RT()>

/*****
*
*   P20_Enzyme_2_RTM()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P20_Enzyme_2_RTM()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C20_Enzyme_2_RTM;
    int nRet;           // Funct. return value
    int nFileIndex = 0;  // Current File Index
    int nLoopCount = 0;  // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

```

```

// Procedure loop.
pPC->nProcLimit = 1;
for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 3, "UseEnzyme2", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 0 );

    // Create/Update the Time Marker
    MSL_TimeMarkerCreate( pPC->szName );

} //<End Procedure Loop>

return 0;
} //<End P20_Enzyme_2_RTM()>

/*****
*
*   P21_Enzyme__2_to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P21_Enzyme__2_to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C21_Enzyme__2_to_MALDI26;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.

```

```

MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 13, "E2", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate Enzyme #2
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 12, 1.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 366, &M2100_Aspirate_Enzyme__2, 1,
        MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume

```

id

```

        dDspVol,          // dispense back volume
        0.0 );           // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }    // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Enzyme #2
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 12, 1.0 );

nRet = EGS_DspEx( pPC, 367, &M2101_Dispense_Enzyme__2, 0,
        dDspVol,          // dispense volume
        3.0,              // post air volume
        1,                // dispense mode
        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) );    // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 368, &M2102_AqueousFlush, 0,
        dDspVol,          // system liquid volume
        5.0,              // post-wash system air gap
        1 );              // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 371, &M2103_AqueousWash, 0,
        dDspVol,          // system liquid volume
        5.0,              // post-wash system air gap
        1 );              // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;    // Abort the test.
if( nRet == 2 ) return 0;    // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

```

```

} //<End P21_Enzyme__2_to_MALDI26()>

/*****
*
*   P22_Enzyme_2_IT()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P22_Enzyme_2_IT()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C22_Enzyme_2_IT;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    MP2_TIME stTime;    // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;
    MSL_StringToTime( "0 0:03:00", &stTime );

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Enzyme_2_RTM", &stTime, NULL, 0 );
        if( nRet == -1 ) return -1;
    } //<End Procedure Loop>

    return 0;
} //<End P22_Enzyme_2_IT()>

/*****
*
*   P23_Buffer__3__E2__to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P23_Buffer__3__E2__to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C23_Buffer__3__E2__to_MALDI26;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops

```



```

int  bDone;           // Last procedure loop flag
int  bEOM             = 0; // End of map indicator
int  nXfrGroups;      // Nbr of transfer groups constructed for the sample
int  nMask;           // General purpose mask
double dMixVol;       // Mix volume (µL)
double dAspVol;       // Aspirate volume (µL)
double dDspVol;       // Dispense volume (µL)
double dXfrVol;       // Total transfer volume (µL)
double dWasteVol = 0;  // Waste volume (µL)
double dBlowoutVol = 0; // Blowout volume (µL)
double dPostAir;      // Post-step aspirate air gap (µL)
int  nDsp;            // Dispense counter
int  nDspMax;         // Maximum number of dispenses
int  nDspMode;        // Dispense mode
double dLLSAspHeight; // Liq. Level Sense Aspirate Height
double dLLSDspHeight; // Liq. Level Sense Dispense Height
double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
double dRetractFromLiqHeight; // Retract from liquid height
double dRetractFromLiqSpeed; // Retract from liquid speed

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 22, "B5", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;

```

```

pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate Buffer #2
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 21, 10.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 378, &M2300_Aspirate_Buffer_2, 1,
                  MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
id
                  dAspVol, // aspirate volume
                  dBlowoutVol, // pre-aspirate air volume (for blowout)
                  3.0, // post-aspirate air volume
                  dDspVol, // dispense back volume
                  0.0 ); // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Buffer #2
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 21, 10.0 );

nRet = EGS_DspEx( pPC, 379, &M2301_Dispense_Buffer_2, 0,
                  dDspVol, // dispense volume
                  3.0, // post air volume
                  1, // dispense mode
                  PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 380, &M2302_AqueousFlush, 0,
                      dDspVol, // system liquid volume
                      5.0, // post-wash system air gap
                      1 ); // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 383, &M2303_AqueousWash, 0,
                      dDspVol, // system liquid volume
                      5.0, // post-wash system air gap
                      1 ); // wash cycles
if( nRet == -1 ) return nRet;

```

```

    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1;    // Abort the test.
    if( nRet == 2 ) return 0;    // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

} //<End P23_Buffer__3__E2__to_MALDI26()>

/*****
*
*   P24_Buffer__4__E2__to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P24_Buffer__4__E2__to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C24_Buffer__4__E2__to_MALDI26;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    int bEOM = 0;           // End of map indicator
    int nXfrGroups;          // Nbr of transfer groups constructed for the sample
    int nMask;               // General purpose mask
    double dMixVol;          // Mix volume (µL)
    double dAspVol;          // Aspirate volume (µL)
    double dDspVol;          // Dispense volume (µL)
    double dXfrVol;          // Total transfer volume (µL)
    double dWasteVol = 0;    // Waste volume (µL)
    double dBlowoutVol = 0;  // Blowout volume (µL)
    double dPostAir;         // Post-step aspirate air gap (µL)
    int nDsp;                // Dispense counter
    int nDspMax;             // Maximum number of dispenses
    int nDspMode;            // Dispense mode
    double dLLSAspHeight;    // Liq. Level Sense Aspirate Height
    double dLLSDspHeight;    // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );

```

```

dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 24, "B6", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate Buffer #2
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 23, 10.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 388, &M2400_Aspirate_Buffer_2, 1,
        MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume
        dDspVol, // dispense back volume
        0.0 ); // No waste
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed
    if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
    // Specify additional aspirate details...

```

```

    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
    PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
    dLLSCLotDetHeight, 1 );

    //-----
    // Dispense Buffer #2
    //-----
    dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 23, 10.0 );

    nRet = EGS_DspEx( pPC, 389, &M2401_Dispense_Buffer__2, 0,
                    dDspVol, // dispense volume
                    3.0, // post air volume
                    1, // dispense mode
                    PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    // Specify additional dispense details...
    EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
    pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

    //-----
    // AqueousFlush
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 390, &M2402_AqueousFlush, 0,
                        dDspVol, // system liquid volume
                        5.0, // post-wash system air gap
                        1 ); // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // AqueousWash
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 393, &M2403_AqueousWash, 0,
                        dDspVol, // system liquid volume
                        5.0, // post-wash system air gap
                        1 ); // wash cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1; // Abort the test.
    if( nRet == 2 ) return 0; // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1; // Abort the test.

return 0;

} //<End P24_Buffer__4__E2__to_MALDI26()>

/*****
*
* P25_Enzyme_2_RT()
*
* <Reserved for text entered into the procedure's comment page.>
*
*****/

```

```

*****/

int P25_Enzyme_2_RT()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C25_Enzyme_2_RT;
    int nRet;           // Funct. return value
    int nFileIndex = 0;  // Current File Index
    int nLoopCount = 0;  // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    MP2_TIME stTime;     // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Enzyme_2_RTM", MSL_GetRtFileColTime( &RtFile3, nFileIndex, 15, &stTime
), NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;
} //<End P25_Enzyme_2_RT()>

/*****
*
*   P26_Enzyme_3_RTM()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P26_Enzyme_3_RTM()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C26_Enzyme_3_RTM;
    int nRet;           // Funct. return value
    int nFileIndex = 0;  // Current File Index
    int nLoopCount = 0;  // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

```

```

// Procedure loop.
pPC->nProcLimit = 1;
for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 4, "UseEnzyme3", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 0 );

    // Create/Update the Time Marker
    MSL_TimeMarkerCreate( pPC->szName );

} //<End Procedure Loop>

return 0;

} //<End P26_Enzyme_3_RTM()>

/*****
*
*   P27_Enzyme__3_to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P27_Enzyme__3_to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C27_Enzyme__3_to_MALDI26;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSclotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.

```

```

MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 15, "E3", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate Enzyme #3
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 14, 1.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 402, &M2700_Aspirate_Enzyme__3, 1,
        MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume

```

id


```

        dDspVol,          // dispense back volume
        0.0 );           // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }    // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Enzyme #3
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 14, 1.0 );

nRet = EGS_DspEx( pPC, 403, &M2701_Dispense_Enzyme__3, 0,
        dDspVol,          // dispense volume
        3.0,             // post air volume
        1,               // dispense mode
        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) );           // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 404, &M2702_AqueousFlush, 0,
        dDspVol,          // system liquid volume
        5.0,             // post-wash system air gap
        1 );             // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 407, &M2703_AqueousWash, 0,
        dDspVol,          // system liquid volume
        5.0,             // post-wash system air gap
        1 );             // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;    // Abort the test.
if( nRet == 2 ) return 0;     // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

```

```

} //<End P27_Enzyme__3_to_MALDI26()>

/*****
*
*   P28_Enzyme_3_IT()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P28_Enzyme_3_IT()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C28_Enzyme_3_IT;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    MP2_TIME stTime;    // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;
    MSL_StringToTime( "0 0:03:00", &stTime );

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Enzyme_3_RTM", &stTime, NULL, 0 );
        if( nRet == -1 ) return -1;
    } //<End Procedure Loop>

    return 0;
} //<End P28_Enzyme_3_IT()>

/*****
*
*   P29_Buffer__3__E3__to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P29_Buffer__3__E3__to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C29_Buffer__3__E3__to_MALDI26;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops

```

```

int  bDone;           // Last procedure loop flag
int  bEOM             = 0; // End of map indicator
int  nXfrGroups;      // Nbr of transfer groups constructed for the sample
int  nMask;           // General purpose mask
double dMixVol;       // Mix volume (µL)
double dAspVol;       // Aspirate volume (µL)
double dDspVol;       // Dispense volume (µL)
double dXfrVol;       // Total transfer volume (µL)
double dWasteVol = 0; // Waste volume (µL)
double dBlowoutVol = 0; // Blowout volume (µL)
double dPostAir;      // Post-step aspirate air gap (µL)
int  nDsp;            // Dispense counter
int  nDspMax;         // Maximum number of dispenses
int  nDspMode;        // Dispense mode
double dLLSAspHeight; // Liq. Level Sense Aspirate Height
double dLLSDspHeight; // Liq. Level Sense Dispense Height
double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
double dRetractFromLiqHeight; // Retract from liquid height
double dRetractFromLiqSpeed; // Retract from liquid speed

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 22, "B7", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;

```

```

pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate Buffer #2
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 21, 10.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 414, &M2900_Aspirate_Buffer_2, 1,
                  MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
id
                  dAspVol, // aspirate volume
                  dBlowoutVol, // pre-aspirate air volume (for blowout)
                  3.0, // post-aspirate air volume
                  dDspVol, // dispense back volume
                  0.0 ); // No waste

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Buffer #2
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 21, 10.0 );

nRet = EGS_DspEx( pPC, 415, &M2901_Dispense_Buffer_2, 0,
                  dDspVol, // dispense volume
                  3.0, // post air volume
                  1, // dispense mode
                  PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 416, &M2902_AqueousFlush, 0,
                      dDspVol, // system liquid volume
                      5.0, // post-wash system air gap
                      1 ); // flush cycles

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 419, &M2903_AqueousWash, 0,
                      dDspVol, // system liquid volume
                      5.0, // post-wash system air gap
                      1 ); // wash cycles

if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

```

```

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;    // Abort the test.
if( nRet == 2 ) return 0;    // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

} //<End P29_Buffer__3__E3__to_MALDI26()>

/*****
*
*   P2A_Buffer__4__E3__to_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P2A_Buffer__4__E3__to_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C2A_Buffer__4__E3__to_MALDI26;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    int bEOM = 0;            // End of map indicator
    int nXfrGroups;          // Nbr of transfer groups constructed for the sample
    int nMask;               // General purpose mask
    double dMixVol;           // Mix volume (µL)
    double dAspVol;           // Aspirate volume (µL)
    double dDspVol;           // Dispense volume (µL)
    double dXfrVol;           // Total transfer volume (µL)
    double dWasteVol = 0;     // Waste volume (µL)
    double dBlowoutVol = 0;   // Blowout volume (µL)
    double dPostAir;          // Post-step aspirate air gap (µL)
    int nDsp;                 // Dispense counter
    int nDspMax;              // Maximum number of dispenses
    int nDspMode;             // Dispense mode
    double dLLSAspHeight;     // Liq. Level Sense Aspirate Height
    double dLLSDspHeight;     // Liq. Level Sense Dispense Height
    double dLLSclotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSclotDetHeight = PrfFileGetLLSclotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );

```

```

dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 24, "B8", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate Buffer #2
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 23, 10.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 424, &M2A00_Aspirate_Buffer_2, 1,
        MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume
        dDspVol, // dispense back volume
        0.0 ); // No waste
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed
    if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
    // Specify additional aspirate details...

```

```

    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
    PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
    dLLSCLotDetHeight, 1 );

    //-----
    // Dispense Buffer #2
    //-----
    dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 23, 10.0 );

    nRet = EGS_DspEx( pPC, 425, &M2A01_Dispense_Buffer__2, 0,
                    dDspVol, // dispense volume
                    3.0, // post air volume
                    1, // dispense mode
                    PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    // Specify additional dispense details...
    EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
    pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

    //-----
    // AqueousFlush
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 426, &M2A02_AqueousFlush, 0,
                        dDspVol, // system liquid volume
                        5.0, // post-wash system air gap
                        1 ); // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // AqueousWash
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 429, &M2A03_AqueousWash, 0,
                        dDspVol, // system liquid volume
                        5.0, // post-wash system air gap
                        1 ); // wash cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1; // Abort the test.
    if( nRet == 2 ) return 0; // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1; // Abort the test.

return 0;

} //<End P2A_Buffer__4_E3__to_MALDI26()>

/*****
*
* P2B_Enzyme_3_RT()
*
* <Reserved for text entered into the procedure's comment page.>
*/

```

```

*
*****/

int P2B_Enzyme_3_RT()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C2B_Enzyme_3_RT;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    MP2_TIME stTime;    // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Enzyme_3_RTM", MSL_GetRtFileColTime( &RtFile3, nFileIndex, 16, &stTime
), NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;
} //<End P2B_Enzyme_3_RT()>

/*****
*
*   P2C_Sample_Matrix_Time_Marker()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P2C_Sample_Matrix_Time_Marker()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C2C_Sample_Matrix_Time_Marker;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

```



```

// Procedure loop.
pPC->nProcLimit = 1;
for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 5, "UseMultipleMatrixAddition", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 0 );

    // Create/Update the Time Marker
    MSL_TimeMarkerCreate( pPC->szName );

} //<End Procedure Loop>

return 0;

} //<End P2C_Sample_Matrix_Time_Marker(>

/*****
*
*   P2D_DHB_Matrix_to_Samples_on_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P2D_DHB_Matrix_to_Samples_on_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C2D_DHB_Matrix_to_Samples_on_MALDI26;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

```

```

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 19, "DHB", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate DHB Matrix for Samples
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 16, 1.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 438, &M2D00_Aspirate_DHB_Matrix_for_Samples, 0,
        MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume
        dDspVol, // dispense back volume
        0.0 ); // No waste

```

id

```

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }    // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed
    if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
    // Specify additional aspirate details...
    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

    //-----
    // Dispense DHB Matrix for Samples
    //-----
    dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 16, 1.0 );

    nRet = EGS_DspEx( pPC, 439, &M2D01_Dispense_DHB_Matrix_for_Samples, 0,
                    dDspVol,          // dispense volume
                    3.0,              // post air volume
                    1,                // dispense mode
                    PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) );          // Blowout delay
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    // Specify additional dispense details...
    EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

    //-----
    // AqueousFlush before OrganicWash-LVT
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 440, &M2D02_AqueousFlush_before_OrganicWash_LVT, 0,
                        dDspVol,          // system liquid volume
                        5.0,              // post-wash system air gap
                        1 );              // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // AqueousWash before OrganicWash-LVT
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 443, &M2D03_AqueousWash_before_OrganicWash_LVT, 0,
                        dDspVol,          // system liquid volume
                        5.0,              // post-wash system air gap
                        1 );              // wash cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // Aspirate OrganicWash-LVT
    //-----
    dAspVol = 10.0;

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 446, &M2D04_Aspirate_OrganicWash_LVT, 0,
                    "OW",          // sample id
                    dAspVol,        // aspirate volume
                    dBlowoutVol,     // pre-aspirate air volume (for blowout)
                    3.0,            // post-aspirate air volume
                    dDspVol,        // dispense back volume
                    0.0 );          // No waste
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }    // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed

```

```

        if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
        // Specify additional aspirate details...
        EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

        //-----
        // Dispense OrganicWash-LVT
        //-----
        dDspVol = 10.0;

        nRet = EGS_DspEx( pPC, 449, &M2D05_Dispense_OrganicWash_LVT, 1,
                        dDspVol,      // dispense volume
                        3.0,          // post air volume
                        1,            // dispense mode
                        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) );      // Blowout delay
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        // Specify additional dispense details...
        EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

        //-----
        // AqueousFlush after OrganicWash-LVT
        //-----
        dDspVol = 2000.0;
        nRet = EGS_PeriWashEx( pPC, 452, &M2D06_AqueousFlush_after_OrganicWash_LVT, 0,
                        dDspVol,      // system liquid volume
                        5.0,          // post-wash system air gap
                        1 );          // flush cycles
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }

        // Specify additional dispense details...
        EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

        //-----
        // AqueousWash after OrganicWash-LVT
        //-----
        dDspVol = 2000.0;
        nRet = EGS_PeriWashEx( pPC, 455, &M2D07_AqueousWash_after_OrganicWash_LVT, 0,
                        dDspVol,      // system liquid volume
                        5.0,          // post-wash system air gap
                        1 );          // wash cycles
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }

        // Specify additional dispense details...
        EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

        // Execute the accumulated sequences only if necessary.
        nRet = EGS_ExecuteEx( pPC, 0 );
        if( nRet == -1 ) return -1;      // Abort the test.
        if( nRet == 2 ) return 0;        // Abort this procedure but not the test.

        // Increment all per-sample maps here (if any)...
        MSL_IncProcSampleMaps( pPC );

    } //<End Procedure Loop>

    // Backout the current sequence if an <End of Map> was reached.
    if( bEOM && (pPC->nSeqCnt > 0) )
        EGS_ClearCurSeq( pPC );

    // Execute any remaining sequences unconditionally.
    nRet = EGS_ExecuteEx( pPC, 1 );
    if( nRet == -1 ) return -1;      // Abort the test.

    return 0;

} //<End P2D_DHB_Matrix_to_Samples_on_MALDI26()>

/*****
*
*   P2E_alpha_Matrix_to_Samples_on_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*/

```

```

*
*****/

int P2E_alpha_Matrix_to_Samples_on_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C2E_alpha_Matrix_to_Samples_on_MALDI26;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (μL)
    double dAspVol; // Aspirate volume (μL)
    double dDspVol; // Dispense volume (μL)
    double dXfrVol; // Total transfer volume (μL)
    double dWasteVol = 0; // Waste volume (μL)
    double dBlowoutVol = 0; // Blowout volume (μL)
    double dPostAir; // Post-step aspirate air gap (μL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 19, "alpha", 1, NULL ) )
        {
            MSL_IncProcSampleMaps( pPC );
            continue;
        }

        // Increment the loop count.
        nLoopCount++;
    }
}

```

```

// Reset the current procedure context.
MSL_SetCurrentProcContext( pPC, 1, 1 );

// Start a dilutor sequence for this sample.
nRet = EGS_NewSequenceEx( pPC );
if( nRet == -1 ) return -1;
pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
if( pPC->hDilutor == -1 ) return -1;
pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate alpha Matrix for Samples
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 16, 1.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 460, &M2E00_Aspirate_alpha_Matrix_for_Samples, 0,
                  MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
id
                  dAspVol, // aspirate volume
                  dBlowoutVol, // pre-aspirate air volume (for blowout)
                  3.0, // post-aspirate air volume
                  dDspVol, // dispense back volume
                  0.0 ); // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense alpha Matrix for Samples
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 16, 1.0 );

nRet = EGS_DspEx( pPC, 461, &M2E01_Dispense_alpha_Matrix_for_Samples, 0,
                  dDspVol, // dispense volume
                  3.0, // post air volume
                  1, // dispense mode
                  PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush before OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 462, &M2E02_AqueousFlush_before_OrganicWash_LVT, 0,
                      dDspVol, // system liquid volume
                      5.0, // post-wash system air gap
                      1 ); // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----

```

```

// AqueousWash before OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 465, &M2E03_AqueousWash_before_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 468, &M2E04_Aspirate_OrganicWash_LVT, 0,
                 "OW",      // sample id
                 dAspVol,    // aspirate volume
                 dBlowoutVol, // pre-aspirate air volume (for blowout)
                 3.0,        // post-aspirate air volume
                 dDspVol,    // dispense back volume
                 0.0 );      // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

//-----
// Dispense OrganicWash-LVT
//-----
dDspVol = 10.0;

nRet = EGS_DspEx( pPC, 471, &M2E05_Dispense_OrganicWash_LVT, 1,
                 dDspVol,      // dispense volume
                 3.0,          // post air volume
                 1,            // dispense mode
                 PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 474, &M2E06_AqueousFlush_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 477, &M2E07_AqueousWash_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles

```

```

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1;    // Abort the test.
    if( nRet == 2 ) return 0;    // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

} //<End P2E_alpha_Matrix_to_Samples_on_MALDI26()>

/*****
*
*   P2F_Sample_Matrix_Timer()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P2F_Sample_Matrix_Timer()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C2F_Sample_Matrix_Timer;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 0, "1", 1, NULL ) )
        {
            MSL_IncProcSampleMaps( pPC );
            continue;
        }

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.

```



```

        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Sample Matrix Time Marker", MSL_CallTimeVarUf( "Uf_stTimePeriod_S2",
&Rt_stTimePeriod_S2, pPC ), NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;

} //<End P2F_Sample_Matrix_Timer()>

/*****
*
*   P30_DHB_Matrix_to_Samples_on_MALDI26___optional()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P30_DHB_Matrix_to_Samples_on_MALDI26___optional()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C30_DHB_Matrix_to_Samples_on_MALDI26___optional;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    int bEOM = 0;            // End of map indicator
    int nXfrGroups;          // Nbr of transfer groups constructed for the sample
    int nMask;               // General purpose mask
    double dMixVol;          // Mix volume (µL)
    double dAspVol;          // Aspirate volume (µL)
    double dDspVol;          // Dispense volume (µL)
    double dXfrVol;          // Total transfer volume (µL)
    double dWasteVol = 0;    // Waste volume (µL)
    double dBlowoutVol = 0;  // Blowout volume (µL)
    double dPostAir;         // Post-step aspirate air gap (µL)
    int nDsp;                // Dispense counter
    int nDspMax;             // Maximum number of dispenses
    int nDspMode;            // Dispense mode
    double dLLSAspHeight;    // Liq. Level Sense Aspirate Height
    double dLLSDspHeight;    // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )

```

```

{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 18, "DHB_o", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate Matrix for Samples - optional
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 17, 1.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 484, &M3000_Aspirate_Matrix_for_Samples___optional, 0,
        MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
id
        dAspVol, // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0, // post-aspirate air volume
        dDspVol, // dispense back volume
        0.0 ); // No waste
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed
    if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
    // Specify additional aspirate details...
    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
    PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
    dLLSCLotDetHeight, 1 );

    //-----
    // Dispense Matrix for Samples - optional
    //-----
    dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 17, 1.0 );

    nRet = EGS_DspEx( pPC, 485, &M3001_Dispende_Matrix_for_Samples___optional, 0,
        dDspVol, // dispense volume
        3.0, // post air volume
        1, // dispense mode
        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
    if( nRet == -1 ) return nRet;

```

```

    if( nRet == -2 ) { bEOM = 1; break; }
    // Specify additional dispense details...
    EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

    //-----
    // AqueousFlush before OrganicWash-LVT
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 486, &M3002_AqueousFlush_before_OrganicWash_LVT, 0,
                          dDspVol, // system liquid volume
                          5.0, // post-wash system air gap
                          1 ); // flush cycles

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // AqueousWash before OrganicWash-LVT
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 489, &M3003_AqueousWash_before_OrganicWash_LVT, 0,
                          dDspVol, // system liquid volume
                          5.0, // post-wash system air gap
                          1 ); // wash cycles

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // Aspirate OrganicWash-LVT
    //-----
    dAspVol = 10.0;

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 492, &M3004_Aspirate_OrganicWash_LVT, 0,
                     "OW", // sample id
                     dAspVol, // aspirate volume
                     dBlowoutVol, // pre-aspirate air volume (for blowout)
                     3.0, // post-aspirate air volume
                     dDspVol, // dispense back volume
                     0.0 ); // No waste

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed
    if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
    // Specify additional aspirate details...
    EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

    //-----
    // Dispense OrganicWash-LVT
    //-----
    dDspVol = 10.0;

    nRet = EGS_DspEx( pPC, 495, &M3005_Dispense_OrganicWash_LVT, 1,
                     dDspVol, // dispense volume
                     3.0, // post air volume
                     1, // dispense mode
                     PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    // Specify additional dispense details...
    EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

    //-----
    // AqueousFlush after OrganicWash-LVT

```

```

//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 498, &M3006_AqueousFlush_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 501, &M3007_AqueousWash_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1;      // Abort the test.
if( nRet == 2 ) return 0;        // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;      // Abort the test.

return 0;

} //<End P30_DHB_Matrix_to_Samples_on_MALDI26___optional()>

/*****
*
*   P31_alpha_Matrix_to_Samples_on_MALDI26___optional()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P31_alpha_Matrix_to_Samples_on_MALDI26___optional()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C31_alpha_Matrix_to_Samples_on_MALDI26___optional;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    int bEOM = 0;            // End of map indicator
    int nXfrGroups;          // Nbr of transfer groups constructed for the sample
    int nMask;               // General purpose mask
    double dMixVol;           // Mix volume (µL)
    double dAspVol;           // Aspirate volume (µL)
    double dDspVol;           // Dispense volume (µL)
    double dXfrVol;           // Total transfer volume (µL)
    double dWasteVol = 0;     // Waste volume (µL)
    double dBlowoutVol = 0;   // Blowout volume (µL)
    double dPostAir;          // Post-step aspirate air gap (µL)
    int nDsp;                 // Dispense counter
    int nDspMax;              // Maximum number of dispenses
    int nDspMode;             // Dispense mode

```

```

double dLLSAspHeight;    // Liq. Level Sense Aspirate Height
double dLLSDspHeight;    // Liq. Level Sense Dispense Height
double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
double dRetractFromLiqHeight; // Retract from liquid height
double dRetractFromLiqSpeed; // Retract from liquid speed

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "Y-Y-Y-Y" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = MSL_GetRtFileColRecords( &RtFile1, 1 );
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile1, nFileIndex, 18, "alpha_o", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate alpha Matrix for Samples
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 17, 1.0 );

```

```

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 506, &M3100_Aspirate_alpha_Matrix_for_Samples, 0,
    MSL_GetRtFileColString( &RtFile1, nFileIndex, 2, "SRC%0000" ), // sample
id
    dAspVol, // aspirate volume
    dBlowoutVol, // pre-aspirate air volume (for blowout)
    3.0, // post-aspirate air volume
    dDspVol, // dispense back volume
    0.0 ); // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense alpha Matrix for Samples
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile1, nFileIndex, 17, 1.0 );

nRet = EGS_DspEx( pPC, 507, &M3101_Dispense_alpha_Matrix_for_Samples, 0,
    dDspVol, // dispense volume
    3.0, // post air volume
    1, // dispense mode
    PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush before OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 508, &M3102_AqueousFlush_before_OrganicWash_LVT, 0,
    dDspVol, // system liquid volume
    5.0, // post-wash system air gap
    1 ); // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash before OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 511, &M3103_AqueousWash_before_OrganicWash_LVT, 0,
    dDspVol, // system liquid volume
    5.0, // post-wash system air gap
    1 ); // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 514, &M3104_Aspirate_OrganicWash_LVT, 0,

```

```

        "OW",      // sample id
        dAspVol,   // aspirate volume
        dBlowoutVol, // pre-aspirate air volume (for blowout)
        3.0,      // post-aspirate air volume
        dDspVol,   // dispense back volume
        0.0 );    // No waste
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed
    if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
    // Specify additional aspirate details...
    EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

    //-----
    // Dispense OrganicWash-LVT
    //-----
    dDspVol = 10.0;

    nRet = EGS_DspEx( pPC, 517, &M3105_Dispense_OrganicWash_LVT, 1,
        dDspVol, // dispense volume
        3.0,     // post air volume
        1,       // dispense mode
        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    // Specify additional dispense details...
    EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

    //-----
    // AqueousFlush after OrganicWash-LVT
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 520, &M3106_AqueousFlush_after_OrganicWash_LVT, 0,
        dDspVol, // system liquid volume
        5.0,    // post-wash system air gap
        1 );    // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // AqueousWash after OrganicWash-LVT
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 523, &M3107_AqueousWash_after_OrganicWash_LVT, 0,
        dDspVol, // system liquid volume
        5.0,    // post-wash system air gap
        1 );    // wash cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1; // Abort the test.
    if( nRet == 2 ) return 0;   // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );
} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1; // Abort the test.

```

```

    return 0;

} //<End P31_alpha_Matrix_to_Samples_on_MALDI26___optional()>

/*****
*
*   P32_Calibrand_Timer()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P32_Calibrand_Timer()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C32_Calibrand_Timer;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag
    MP2_TIME stTime;    // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;
    MSL_StringToTime( "0 0:05:00", &stTime );

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 17, "C1", 1, NULL ) )
        {
            MSL_IncProcSampleMaps( pPC );
            continue;
        }

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Wait for timer to expire
        nRet = MSL_TimeMarkerWait( "Calibrand Time Marker", &stTime, NULL, 0 );
        if( nRet == -1 ) return -1;

    } //<End Procedure Loop>

    return 0;

} //<End P32_Calibrand_Timer()>

/*****
*
*   P33_Calibrand_Matrix_Time_Marker()
*
*****/

```



```

*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P33_Calibrand_Matrix_Time_Marker()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C33_Calibrand_Matrix_Time_Marker;
    int nRet;           // Funct. return value
    int nFileIndex = 0;  // Current File Index
    int nLoopCount = 0;  // Number of unskipped procedure loops
    int bDone;           // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 16, "Use_Calibrand", 1, NULL ) )
        {
            MSL_IncProcSampleMaps( pPC );
            continue;
        }

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Create/Update the Time Marker
        MSL_TimeMarkerCreate( pPC->szName );

    } //<End Procedure Loop>

    return 0;
} //<End P33_Calibrand_Matrix_Time_Marker()>

/*****
*
*   P34_DHB_Matrix_to_Calibrand_on_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P34_DHB_Matrix_to_Calibrand_on_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C34_DHB_Matrix_to_Calibrand_on_MALDI26;
    int nRet;           // Funct. return value
    int nFileIndex = 0;  // Current File Index
    int nLoopCount = 0;  // Number of unskipped procedure loops
    int bDone;           // Last procedure loop flag
    int bEOM = 0;        // End of map indicator
    int nXfrGroups;      // Nbr of transfer groups constructed for the sample
    int nMask;           // General purpose mask
    double dMixVol;       // Mix volume (µL)
    double dAspVol;       // Aspirate volume (µL)
    double dDspVol;       // Dispense volume (µL)

```

```

double dXfrVol;           // Total transfer volume (µL)
double dWasteVol = 0;     // Waste volume (µL)
double dBlowoutVol = 0;   // Blowout volume (µL)
double dPostAir;          // Post-step aspirate air gap (µL)
int nDsp;                 // Dispense counter
int nDspMax;              // Maximum number of dispenses
int nDspMode;             // Dispense mode
double dLLSAspHeight;     // Liq. Level Sense Aspirate Height
double dLLSDspHeight;     // Liq. Level Sense Dispense Height
double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
double dRetractFromLiqHeight; // Retract from liquid height
double dRetractFromLiqSpeed; // Retract from liquid speed

// Set the current procedure context.
MSL_SetCurrentProcContext( pPC, 0, 0 );

// Reset all well maps that require it at the start of a procedure.
MSL_ResetProcMaps( pPC );

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "N-Y-N-N" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile2, nFileIndex, 19, "DHB", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;
    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.

```

```

EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate Matrix for Calibrant
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 16, 1.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 532, &M3400_Aspirate_Matrix_for_Calibrant, 1,
    "Call", // sample id
    dAspVol, // aspirate volume
    dBlowoutVol, // pre-aspirate air volume (for blowout)
    3.0, // post-aspirate air volume
    dDspVol, // dispense back volume
    0.0 ); // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Matrix for Calibrant
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 16, 1.0 );

nRet = EGS_DspEx( pPC, 533, &M3401_Dispende_Matrix_for_Calibrant, 0,
    dDspVol, // dispense volume
    3.0, // post air volume
    1, // dispense mode
    PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush before OrganicWash-LVT_1
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 534, &M3402_AqueousFlush_before_OrganicWash_LVT_1, 0,
    dDspVol, // system liquid volume
    5.0, // post-wash system air gap
    1 ); // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash before OrganicWash-LVT_1
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 537, &M3403_AqueousWash_before_OrganicWash_LVT_1, 0,
    dDspVol, // system liquid volume
    5.0, // post-wash system air gap
    1 ); // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

```

```

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 540, &M3404_Aspirate_OrganicWash_LVT, 0,
                  "OW",      // sample id
                  dAspVol,    // aspirate volume
                  dBlowoutVol, // pre-aspirate air volume (for blowout)
                  3.0,        // post-aspirate air volume
                  dDspVol,     // dispense back volume
                  0.0 );      // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

//-----
// Dispense OrganicWash-LVT
//-----
dDspVol = 10.0;

nRet = EGS_DspEx( pPC, 543, &M3405_Dispense_OrganicWash_LVT, 1,
                  dDspVol,    // dispense volume
                  3.0,        // post air volume
                  1,          // dispense mode
                  PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 546, &M3406_AqueousFlush_after_OrganicWash_LVT, 0,
                      dDspVol,    // system liquid volume
                      5.0,        // post-wash system air gap
                      1 );        // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 549, &M3407_AqueousWash_after_OrganicWash_LVT, 0,
                      dDspVol,    // system liquid volume
                      5.0,        // post-wash system air gap
                      1 );        // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1; // Abort the test.
if( nRet == 2 ) return 0;   // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.

```

```

    if( bEOM && (pPC->nSeqCnt > 0) )
        EGS_ClearCurSeq( pPC );

    // Execute any remaining sequences unconditionally.
    nRet = EGS_ExecuteEx( pPC, 1 );
    if( nRet == -1 ) return -1;    // Abort the test.

    return 0;
} //<End P34_DHB_Matrix_to_Calibrand_on_MALDI26()>

/*****
*
*   P35_alpha_Matrix_to_Calibrand_on_MALDI26()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P35_alpha_Matrix_to_Calibrand_on_MALDI26()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C35_alpha_Matrix_to_Calibrand_on_MALDI26;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    int bEOM = 0;            // End of map indicator
    int nXfrGroups;          // Nbr of transfer groups constructed for the sample
    int nMask;               // General purpose mask
    double dMixVol;           // Mix volume (µL)
    double dAspVol;           // Aspirate volume (µL)
    double dDspVol;           // Dispense volume (µL)
    double dXfrVol;           // Total transfer volume (µL)
    double dWasteVol = 0;     // Waste volume (µL)
    double dBlowoutVol = 0;   // Blowout volume (µL)
    double dPostAir;          // Post-step aspirate air gap (µL)
    int nDsp;                 // Dispense counter
    int nDspMax;              // Maximum number of dispenses
    int nDspMode;             // Dispense mode
    double dLLSAspHeight;     // Liq. Level Sense Aspirate Height
    double dLLSDspHeight;     // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "N-Y-N-N" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = 1;
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;
    }
}

```

```

// If at or past the last loop, set the done flag.
bDone = (pPC->nProcLoop >= pPC->nProcLimit);

// Child procedure calls would be written here...

// If done, break now.
if( bDone ) break;

// Should this loop be skipped?
if( !MSL_GetRtFileColBoolByFlag( &RtFile2, nFileIndex, 19, "alpha", 1, NULL ) )
{
    MSL_IncProcSampleMaps( pPC );
    continue;
}

// Increment the loop count.
nLoopCount++;

// Reset the current procedure context.
MSL_SetCurrentProcContext( pPC, 1, 1 );

// Start a dilutor sequence for this sample.
nRet = EGS_NewSequenceEx( pPC );
if( nRet == -1 ) return -1;
pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
if( pPC->hDilutor == -1 ) return -1;
pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate Matrix for Calibrand
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 16, 1.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 554, &M3500_Aspirate_Matrix_for_Calibrand, 1,
                "Call",      // sample id
                dAspVol,      // aspirate volume
                dBlowoutVol,   // pre-aspirate air volume (for blowout)
                3.0,          // post-aspirate air volume
                dDspVol,       // dispense back volume
                0.0 );         // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSclotDetHeight, 1 );

//-----
// Dispense Matrix for Calibrand
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 16, 1.0 );

nRet = EGS_DspEx( pPC, 555, &M3501_Dispense_Matrix_for_Calibrand, 0,
                dDspVol,      // dispense volume
                3.0,          // post air volume
                1,            // dispense mode
                PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...

```

```

EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush before OrganicWash-LVT_1
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 556, &M3502_AqueousFlush_before_OrganicWash_LVT_1, 0,
                        dDspVol,      // system liquid volume
                        5.0,          // post-wash system air gap
                        1 );          // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash before OrganicWash-LVT_1
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 559, &M3503_AqueousWash_before_OrganicWash_LVT_1, 0,
                        dDspVol,      // system liquid volume
                        5.0,          // post-wash system air gap
                        1 );          // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 562, &M3504_Aspirate_OrganicWash_LVT, 0,
                  "OW",      // sample id
                  dAspVol,    // aspirate volume
                  dBlowoutVol, // pre-aspirate air volume (for blowout)
                  3.0,        // post-aspirate air volume
                  dDspVol,    // dispense back volume
                  0.0 );      // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

//-----
// Dispense OrganicWash-LVT
//-----
dDspVol = 10.0;

nRet = EGS_DspEx( pPC, 565, &M3505_Dispense_OrganicWash_LVT, 1,
                  dDspVol,      // dispense volume
                  3.0,          // post air volume
                  1,            // dispense mode
                  PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush after OrganicWash-LVT
//-----
dDspVol = 2000.0;

```

```

nRet = EGS_PeriWashEx( pPC, 568, &M3506_AqueousFlush_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash after OrganicWash-LVT
//-----

dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 571, &M3507_AqueousWash_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

// Execute the accumulated sequences only if necessary.
nRet = EGS_ExecuteEx( pPC, 0 );
if( nRet == -1 ) return -1; // Abort the test.
if( nRet == 2 ) return 0;   // Abort this procedure but not the test.

// Increment all per-sample maps here (if any)...
MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1; // Abort the test.

return 0;

} //<End P35_alpha_Matrix_to_Calibrant_on_MALDI26()>

/*****
*
*   P36_Calibrant_Matrix_Timer()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P36_Calibrant_Matrix_Timer()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C36_Calibrant_Matrix_Timer;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    MP2_TIME stTime; // Time Period To Pause

    // Setup the time period structure
    stTime.bPeriod = 1;
    MSL_StringToTime( "0 0:03:00", &stTime );

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.

```



```

pPC->nProcLimit = 1;
for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 0, "1", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 0 );

    // Wait for timer to expire
    nRet = MSL_TimeMarkerWait( "Calibrant Matrix Time Marker", &stTime, NULL, 0 );
    if( nRet == -1 ) return -1;

} //<End Procedure Loop>

return 0;

} //<End P36_Calibrant_Matrix_Timer()>

/*****
*
*   P37_DHB_Matrix_to_Calibrant_on_MALDI26___optional()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P37_DHB_Matrix_to_Calibrant_on_MALDI26___optional()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C37_DHB_Matrix_to_Calibrant_on_MALDI26___optional;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSclotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

```

```

// Set the performance file.
pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
EGS_SetLiquidType( "SerumBlowoutFT.prf" );

// Load the non-volume dependent performance parameters.
dLLSAspHeight      = PrfFileGetLLSAspHeight( pPC->hPrfFile );
dLLSDspHeight      = PrfFileGetLLSDspHeight( pPC->hPrfFile );
dLLSCLotDetHeight  = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
dRetractFromLiqSpeed  = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

// Toplevel procedures start with dilutor number 1.
EGS_SetNextDilutor( 1 );

// Load list of dilutor in use flags
strcpy( pPC->szDilInUse, "N-Y-N-N" );
pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

// Procedure sample loop.
pPC->nSamples = 1;
pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
{
    nFileIndex = pPC->nProcLoop;

    // If at or past the last loop, set the done flag.
    bDone = (pPC->nProcLoop >= pPC->nProcLimit);

    // Child procedure calls would be written here...

    // If done, break now.
    if( bDone ) break;

    // Should this loop be skipped?
    if( !MSL_GetRtFileColBoolByFlag( &RtFile2, nFileIndex, 18, "DHB_o", 1, NULL ) )
    {
        MSL_IncProcSampleMaps( pPC );
        continue;
    }

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 1 );

    // Start a dilutor sequence for this sample.
    nRet = EGS_NewSequenceEx( pPC );
    if( nRet == -1 ) return -1;
    pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
    if( pPC->hDilutor == -1 ) return -1;
    pPC->nSeqCnt++;

    nXfrGroups = 0;

    // Set LLS Verification flag.
    EGS_SetLLSVerificationMode( 0 );

    // Load the pipette rule.
    EGS_SetPipetteRule( 0 ); // No Waste

    //-----
    // Aspirate Matrix for Calibrant - optional
    //-----
    dAspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 17, 1.0 );

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 578, &M3700_Aspirate_Matrix_for_Calibrant___optional, 1,
        "Call",          // sample id
        dAspVol,         // aspirate volume
        dBlowoutVol,     // pre-aspirate air volume (for blowout)
        3.0,             // post-aspirate air volume
        dDspVol,         // dispense back volume
        0.0 );           // No waste

```

```

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }    // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed
    if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
    // Specify additional aspirate details...
    EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

    //-----
    // Dispense Matrix for Calibrant - optional
    //-----
    dDspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 17, 1.0 );

    nRet = EGS_DspEx( pPC, 579, &M3701_Dispense_Matrix_for_Calibrant___optional, 0,
                    dDspVol,          // dispense volume
                    3.0,              // post air volume
                    1,                // dispense mode
                    PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) );          // Blowout delay
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    // Specify additional dispense details...
    EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

    //-----
    // AqueousFlush before OrganicWash-LVT_2
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 580, &M3702_AqueousFlush_before_OrganicWash_LVT_2, 0,
                        dDspVol,          // system liquid volume
                        5.0,              // post-wash system air gap
                        1 );              // flush cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // AqueousWash before OrganicWash-LVT_2
    //-----
    dDspVol = 2000.0;
    nRet = EGS_PeriWashEx( pPC, 583, &M3703_AqueousWash_before_OrganicWash_LVT_2, 0,
                        dDspVol,          // system liquid volume
                        5.0,              // post-wash system air gap
                        1 );              // wash cycles
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    //-----
    // Aspirate OrganicWash-LVT
    //-----
    dAspVol = 10.0;

    dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

    dDspVol = 0.0;

    nRet = EGS_AspEx( pPC, 586, &M3704_Aspirate_OrganicWash_LVT, 0,
                    "OW",          // sample id
                    dAspVol,        // aspirate volume
                    dBlowoutVol,    // pre-aspirate air volume (for blowout)
                    3.0,            // post-aspirate air volume
                    dDspVol,        // dispense back volume
                    0.0 );          // No waste
    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }
    if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; }    // Skip
Sample?

    // Don't pickup blowout later if an aspirate was performed

```

```

        if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
        // Specify additional aspirate details...
        EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

        //-----
        // Dispense OrganicWash-LVT
        //-----
        dDspVol = 10.0;

        nRet = EGS_DspEx( pPC, 589, &M3705_Dispense_OrganicWash_LVT, 1,
                        dDspVol,      // dispense volume
                        3.0,          // post air volume
                        1,            // dispense mode
                        PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) );      // Blowout delay
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }
        // Specify additional dispense details...
        EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

        //-----
        // AqueousFlush after OrganicWash-LVT
        //-----
        dDspVol = 2000.0;
        nRet = EGS_PeriWashEx( pPC, 592, &M3706_AqueousFlush_after_OrganicWash_LVT, 0,
                        dDspVol,      // system liquid volume
                        5.0,          // post-wash system air gap
                        1 );          // flush cycles
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }

        // Specify additional dispense details...
        EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

        //-----
        // AqueousWash after OrganicWash-LVT
        //-----
        dDspVol = 2000.0;
        nRet = EGS_PeriWashEx( pPC, 595, &M3707_AqueousWash_after_OrganicWash_LVT, 0,
                        dDspVol,      // system liquid volume
                        5.0,          // post-wash system air gap
                        1 );          // wash cycles
        if( nRet == -1 ) return nRet;
        if( nRet == -2 ) { bEOM = 1; break; }

        // Specify additional dispense details...
        EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

        // Execute the accumulated sequences only if necessary.
        nRet = EGS_ExecuteEx( pPC, 0 );
        if( nRet == -1 ) return -1;      // Abort the test.
        if( nRet == 2 ) return 0;        // Abort this procedure but not the test.

        // Increment all per-sample maps here (if any)...
        MSL_IncProcSampleMaps( pPC );

    } //<End Procedure Loop>

    // Backout the current sequence if an <End of Map> was reached.
    if( bEOM && (pPC->nSeqCnt > 0) )
        EGS_ClearCurSeq( pPC );

    // Execute any remaining sequences unconditionally.
    nRet = EGS_ExecuteEx( pPC, 1 );
    if( nRet == -1 ) return -1;      // Abort the test.

    return 0;

} //<End P37_DHB_Matrix_to_Calibrant_on_MALDI26___optional()>

/*****
*
*   P38_alpha_Matrix_to_Calibrant_on_MALDI26___optional()
*
*****/

```

```

*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P38_alpha_Matrix_to_Calibrant_on_MALDI26___optional()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C38_alpha_Matrix_to_Calibrant_on_MALDI26___optional;
    int nRet; // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone; // Last procedure loop flag
    int bEOM = 0; // End of map indicator
    int nXfrGroups; // Nbr of transfer groups constructed for the sample
    int nMask; // General purpose mask
    double dMixVol; // Mix volume (µL)
    double dAspVol; // Aspirate volume (µL)
    double dDspVol; // Dispense volume (µL)
    double dXfrVol; // Total transfer volume (µL)
    double dWasteVol = 0; // Waste volume (µL)
    double dBlowoutVol = 0; // Blowout volume (µL)
    double dPostAir; // Post-step aspirate air gap (µL)
    int nDsp; // Dispense counter
    int nDspMax; // Maximum number of dispenses
    int nDspMode; // Dispense mode
    double dLLSAspHeight; // Liq. Level Sense Aspirate Height
    double dLLSDspHeight; // Liq. Level Sense Dispense Height
    double dLLSCLotDetHeight; // Liq. Level Sense Clot Detect Height
    double dRetractFromLiqHeight; // Retract from liquid height
    double dRetractFromLiqSpeed; // Retract from liquid speed

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Reset all well maps that require it at the start of a procedure.
    MSL_ResetProcMaps( pPC );

    // Set the performance file.
    pPC->hPrfFile = PrfFileOpen( "SerumBlowoutFT.prf" );
    EGS_SetLiquidType( "SerumBlowoutFT.prf" );

    // Load the non-volume dependent performance parameters.
    dLLSAspHeight = PrfFileGetLLSAspHeight( pPC->hPrfFile );
    dLLSDspHeight = PrfFileGetLLSDspHeight( pPC->hPrfFile );
    dLLSCLotDetHeight = PrfFileGetLLSCLotDetHeight( pPC->hPrfFile );
    dRetractFromLiqHeight = PrfFileGetRetractFromLiqHeight( pPC->hPrfFile );
    dRetractFromLiqSpeed = PrfFileGetRetractFromLiqSpeed( pPC->hPrfFile );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Load list of dilutor in use flags
    strcpy( pPC->szDilInUse, "N-Y-N-N" );
    pPC->nDilInUse = EGS_DilutorsInUse( pPC->szDilInUse );

    // Procedure sample loop.
    pPC->nSamples = 1;
    pPC->nProcLimit = pPC->nSamples + pPC->nSampOffset;
    for( pPC->nProcLoop = pPC->nSampOffset; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile2, nFileIndex, 18, "alpha_o", 1, NULL ) )
        {
            MSL_IncProcSampleMaps( pPC );
            continue;
        }

        // Increment the loop count.
    }
}

```

```

nLoopCount++;

// Reset the current procedure context.
MSL_SetCurrentProcContext( pPC, 1, 1 );

// Start a dilutor sequence for this sample.
nRet = EGS_NewSequenceEx( pPC );
if( nRet == -1 ) return -1;
pPC->hDilutor = EGS_GetNextDilutor( pPC->szDilInUse );
if( pPC->hDilutor == -1 ) return -1;
pPC->nSeqCnt++;
nXfrGroups = 0;

// Set LLS Verification flag.
EGS_SetLLSVerificationMode( 0 );

// Load the pipette rule.
EGS_SetPipetteRule( 0 ); // No Waste

//-----
// Aspirate Matrix for Calibrant - optional
//-----
dAspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 17, 1.0 );

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 600, &M3800_Aspirate_Matrix_for_Calibrant___optional, 1,
                  "Call",          // sample id
                  dAspVol,          // aspirate volume
                  dBlowoutVol,      // pre-aspirate air volume (for blowout)
                  3.0,             // post-aspirate air volume
                  dDspVol,          // dispense back volume
                  0.0 );           // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, dLLSAspHeight, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ),
PrfFileGetAspDelay( pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0,
dLLSCLotDetHeight, 1 );

//-----
// Dispense Matrix for Calibrant - optional
//-----
dDspVol = MSL_GetRtFileColDouble( &RtFile2, nFileIndex, 17, 1.0 );

nRet = EGS_DspEx( pPC, 601, &M3801_Dispense_Matrix_for_Calibrant___optional, 0,
                  dDspVol,          // dispense volume
                  3.0,             // post air volume
                  1,               // dispense mode
                  PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 0.25, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush before OrganicWash-LVT_2
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 602, &M3802_AqueousFlush_before_OrganicWash_LVT_2, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----

```

```

// AqueousWash before OrganicWash-LVT_2
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 605, &M3803_AqueousWash_before_OrganicWash_LVT_2, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// Aspirate OrganicWash-LVT
//-----
dAspVol = 10.0;

dBlowoutVol = PrfFileGetBlowoutVolume( pPC->hPrfFile, dAspVol );

dDspVol = 0.0;

nRet = EGS_AspEx( pPC, 608, &M3804_Aspirate_OrganicWash_LVT, 0,
                 "OW",      // sample id
                 dAspVol,    // aspirate volume
                 dBlowoutVol, // pre-aspirate air volume (for blowout)
                 3.0,        // post-aspirate air volume
                 dDspVol,    // dispense back volume
                 0.0 );      // No waste
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
if( nRet == -3 ) { EGS_ClearCurSeq( pPC ); MSL_IncProcSampleMaps( pPC ); continue; } // Skip
Sample?

// Don't pickup blowout later if an aspirate was performed
if( dAspVol > 0.0 ) dBlowoutVol = 0.0;
// Specify additional aspirate details...
EGS_SetAspDetails( 4, 5.0, 1.0, PrfFileGetAspSpeed( pPC->hPrfFile, dAspVol ), PrfFileGetAspDelay(
pPC->hPrfFile, dAspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 9, 0, 0, dLLSCLotDetHeight, 1 );

//-----
// Dispense OrganicWash-LVT
//-----
dDspVol = 10.0;

nRet = EGS_DspEx( pPC, 611, &M3805_Dispense_OrganicWash_LVT, 1,
                 dDspVol,      // dispense volume
                 3.0,          // post air volume
                 1,            // dispense mode
                 PrfFileGetBlowoutDelay( pPC->hPrfFile, dDspVol ) ); // Blowout delay
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }
// Specify additional dispense details...
EGS_SetDspDetails( 0, 0, 1.0, PrfFileGetDspSpeed( pPC->hPrfFile, dDspVol ), PrfFileGetDspDelay(
pPC->hPrfFile, dDspVol ), dRetractFromLiqHeight, dRetractFromLiqSpeed, 0 );

//-----
// AqueousFlush after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 614, &M3806_AqueousFlush_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // flush cycles
if( nRet == -1 ) return nRet;
if( nRet == -2 ) { bEOM = 1; break; }

// Specify additional dispense details...
EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

//-----
// AqueousWash after OrganicWash-LVT
//-----
dDspVol = 2000.0;
nRet = EGS_PeriWashEx( pPC, 617, &M3807_AqueousWash_after_OrganicWash_LVT, 0,
                      dDspVol,      // system liquid volume
                      5.0,          // post-wash system air gap
                      1 );          // wash cycles

```

```

    if( nRet == -1 ) return nRet;
    if( nRet == -2 ) { bEOM = 1; break; }

    // Specify additional dispense details...
    EGS_SetDspDetails( 1, 0, 0.0, -1, 0, dRetractFromLiqHeight, -1, 0 );

    // Execute the accumulated sequences only if necessary.
    nRet = EGS_ExecuteEx( pPC, 0 );
    if( nRet == -1 ) return -1;    // Abort the test.
    if( nRet == 2 ) return 0;    // Abort this procedure but not the test.

    // Increment all per-sample maps here (if any)...
    MSL_IncProcSampleMaps( pPC );

} //<End Procedure Loop>

// Backout the current sequence if an <End of Map> was reached.
if( bEOM && (pPC->nSeqCnt > 0) )
    EGS_ClearCurSeq( pPC );

// Execute any remaining sequences unconditionally.
nRet = EGS_ExecuteEx( pPC, 1 );
if( nRet == -1 ) return -1;    // Abort the test.

return 0;

} //<End P38_alpha_Matrix_to_Calibrant_on_MALDI26___optional()>

/*****
*
*   P39_FTP_Transfer_to_MALDI_Reflex_III()
*
*   Transfers the Sample List via FTP to the MALDI Computer "bruker-sun"
*   Note:
*   This program contains a script, that only works, if the files are located
*   in the appropriate Directories
*
*****/

int P39_FTP_Transfer_to_MALDI_Reflex_III()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C39_FTP_Transfer_to_MALDI_Reflex_III;
    int nRet;                // Funct. return value
    int nFileIndex = 0;      // Current File Index
    int nLoopCount = 0;      // Number of unskipped procedure loops
    int bDone;               // Last procedure loop flag
    char szCmd[256];         // Executable Cmd line

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Should this loop be skipped?
        if( !MSL_GetRtFileColBoolByFlag( &RtFile3, nFileIndex, 18, "FTP", 1, NULL ) )
        {
            MSL_IncProcSampleMaps( pPC );
            continue;
        }
    }
}

```



```

    // Increment the loop count.
    nLoopCount++;

    // Reset the current procedure context.
    MSL_SetCurrentProcContext( pPC, 1, 0 );

    // Make sure string parameters coming from a file are auto allocated
    MSL_SetRtFileStringAutoAlloc( 1 );

    // Invoke User Executable Program...
    sprintf( szCmd, "C:\\Packard\\MultiPROBE\\Robert\\Multiprobe\\user exectable\\ftp\\MALDI2.exe" );
    nRet = MSL_CreateProcess( szCmd, "", 0x1);
    if( nRet == -1 ) return -1;    // Check for failure

    // Release any auto allocated strings.
    MSL_SetRtFileStringAutoAlloc( 0 );

} //<End Procedure Loop>

return 0;

} //<End P39_FTP_Transfer_to_MALDI_Reflex_III()>

/*****
*
*   P3A_User_Information()
*
*   <Reserved for text entered into the procedure's comment page.>
*
*****/

int P3A_User_Information()
{
    MP2_PROC_CONTEXT_DEF *pPC = &C3A_User_Information;
    int nRet;           // Funct. return value
    int nFileIndex = 0; // Current File Index
    int nLoopCount = 0; // Number of unskipped procedure loops
    int bDone;          // Last procedure loop flag

    // Set the current procedure context.
    MSL_SetCurrentProcContext( pPC, 0, 0 );

    // Toplevel procedures start with dilutor number 1.
    EGS_SetNextDilutor( 1 );

    // Procedure loop.
    pPC->nProcLimit = 1;
    for( pPC->nProcLoop = 0; pPC->nProcLoop <= pPC->nProcLimit; pPC->nProcLoop++ )
    {
        nFileIndex = pPC->nProcLoop;

        // If at or past the last loop, set the done flag.
        bDone = (pPC->nProcLoop >= pPC->nProcLimit);

        // Child procedure calls would be written here...

        // If done, break now.
        if( bDone ) break;

        // Increment the loop count.
        nLoopCount++;

        // Reset the current procedure context.
        MSL_SetCurrentProcContext( pPC, 1, 0 );

        // Sound the buzzer for 1/8 second
        MSL_Beep( 500, 0.125 );

        // Display User Message.
        nRet = MSL_MessageDialog( 0,           // No dialog owner
                                "User Information", // Dialog Title
                                "                Close all Tubes and Troughs !" );

        to Prevent a Change of Solvent Composition or Dry-Out !
        ",           // Dialog Message

```

```

        2,      // Icon: Exclamation
        1,      // Btns: OK
        1,      // Default to 1st button
        0 );    // Reserved flags field

    // Test User Reply: 2=>Cancel, 7=>No
    if( (nRet == 2) || (nRet == 7) ) return -1;

} //<End Procedure Loop>

return 0;

} //<End P3A_User_Information()>

/*****
*
*   main()
*
*****/

int main(int argc, char **argv)
{
    int nErr;

    // Set the EGS dump table and single step modes as well as
    // the evaluation level.
    EGS_SetDumpMode( 0 );
    EGS_SetSingleStep( 0 );
    EGS_SetEvaluationLevel( 0 );

    // Initialize runtime variables.
    IniAllRtVariables();

    // Load Instrument Data
    nErr = MSL_CreateMachine(argv[0]);
    if( nErr ) return;

    // Initialize Instrument
    nErr = MSL_InitializeMachine( 0 );
    if( nErr ) return;

    // Initialize file, well map and labware definitions.
    IniAllRtFileDefs();
    IniAllRtSmpListDefs();
    IniAllWellMapDefs();
    IniAllProcDefs();
    IniAllLabwareDefs();

    // Call each 1st level procedure in the outline.
    // Quit test if any procedure returns a failure.
    nErr = P00_Verify_Deck_Layout();
    if( nErr ) return;

    nErr = P01_Danger_of_Test_Failiure_and_LVT_Damage__();
    if( nErr ) return;

    nErr = P02_Pre_Test_Flush_Wash();
    if( nErr ) return;

    nErr = P03_Enzyme_Panel();
    if( nErr ) return;

    nErr = P04_Enzyme_Pre_Incubation();
    if( nErr ) return;

    nErr = P05_Buffer_Panel();
    if( nErr ) return;

    nErr = P06_Matrices_Panel();
    if( nErr ) return;

    nErr = P07_Organic_Wash_Matrices_and_Buffers();
    if( nErr ) return;

    nErr = P08_Reagent_Panel();
    if( nErr ) return;

```

```
nErr = P09_Organic_Wash_Reagent();
if( nErr ) return;

nErr = P0A_Aqueous_Flush_Wash();
if( nErr ) return;

nErr = P0B_Sample_Spotting_Time_Marker();
if( nErr ) return;

nErr = P0C_Samples_to_MALDI26();
if( nErr ) return;

nErr = P0D_Calibrant_Time_Marker();
if( nErr ) return;

nErr = P0E_Calibrant_to_MALDI26();
if( nErr ) return;

nErr = P0F_Sample_Drying_Timer();
if( nErr ) return;

nErr = P10_Buffer__1_Spotting_Time_Marker();
if( nErr ) return;

nErr = P11_Buffer__1_to_MALDI26();
if( nErr ) return;

nErr = P12_Buffer__1_Timer();
if( nErr ) return;

nErr = P13_Reagent_Spotting_Time_Marker();
if( nErr ) return;

nErr = P14_Reagent_to_MALDI26();
if( nErr ) return;

nErr = P15_Reaction_Timer();
if( nErr ) return;

nErr = P16_Buffer__2_Spotting_Time_Marker();
if( nErr ) return;

nErr = P17_Buffer__2_to_MALDI26();
if( nErr ) return;

nErr = P18_Buffer__2_Timer();
if( nErr ) return;

nErr = P19_Enzyme_Pre_Incubation_Time();
if( nErr ) return;

nErr = P1A_Enzyme_1_RTM();
if( nErr ) return;

nErr = P1B_Enzyme__1_to_MALDI26();
if( nErr ) return;

nErr = P1C_Enzyme_1_IT();
if( nErr ) return;

nErr = P1D_Buffer__3__E1__to_MALDI26();
if( nErr ) return;

nErr = P1E_Buffer__4__E1__to_MALDI26();
if( nErr ) return;

nErr = P1F_Enzyme_1_RT();
if( nErr ) return;

nErr = P20_Enzyme_2_RTM();
if( nErr ) return;

nErr = P21_Enzyme__2_to_MALDI26();
if( nErr ) return;

nErr = P22_Enzyme_2_IT();
```

```
if( nErr ) return;

nErr = P23_Buffer__3__E2__to_MALDI26();
if( nErr ) return;

nErr = P24_Buffer__4__E2__to_MALDI26();
if( nErr ) return;

nErr = P25_Enzyme_2_RT();
if( nErr ) return;

nErr = P26_Enzyme_3_RTM();
if( nErr ) return;

nErr = P27_Enzyme__3_to_MALDI26();
if( nErr ) return;

nErr = P28_Enzyme_3_IT();
if( nErr ) return;

nErr = P29_Buffer__3__E3__to_MALDI26();
if( nErr ) return;

nErr = P2A_Buffer__4__E3__to_MALDI26();
if( nErr ) return;

nErr = P2B_Enzyme_3_RT();
if( nErr ) return;

nErr = P2C_Sample_Matrix_Time_Marker();
if( nErr ) return;

nErr = P2D_DHB_Matrix_to_Samples_on_MALDI26();
if( nErr ) return;

nErr = P2E_alpha_Matrix_to_Samples_on_MALDI26();
if( nErr ) return;

nErr = P2F_Sample_Matrix_Timer();
if( nErr ) return;

nErr = P30_DHB_Matrix_to_Samples_on_MALDI26___optional();
if( nErr ) return;

nErr = P31_alpha_Matrix_to_Samples_on_MALDI26___optional();
if( nErr ) return;

nErr = P32_Calibrand_Timer();
if( nErr ) return;

nErr = P33_Calibrand_Matrix_Time_Marker();
if( nErr ) return;

nErr = P34_DHB_Matrix_to_Calibrand_on_MALDI26();
if( nErr ) return;

nErr = P35_alpha_Matrix_to_Calibrand_on_MALDI26();
if( nErr ) return;

nErr = P36_Calibrand_Matrix_Timer();
if( nErr ) return;

nErr = P37_DHB_Matrix_to_Calibrand_on_MALDI26___optional();
if( nErr ) return;

nErr = P38_alpha_Matrix_to_Calibrand_on_MALDI26___optional();
if( nErr ) return;

nErr = P39_FTP_Transfer_to_MALDI_Reflex_III();
if( nErr ) return;

nErr = P3A_User_Information();
if( nErr ) return;

}

// End of Script
```

3 Die Datei "MALDI2.exe – Quellcode"

```
; *****
; MALDI2.exe Quellcode
; =====
; Einträge für Host, Port, User und Password sind aus Sicherheitsgründen hier nicht explizit aufgeführt!
; *****

; ***** First provide the Login information...
Host="*****"
Port="##"
User="*****"
Password="*****"

; ***** Timeout in seconds...
Timeout=120

; ***** ...and then connect to FTP host...
connect

; ***** Command sequence...

remote-cd  "/home/tof/automation/4.2/import"
    Binary
    SendIfNewer "C:\Packard\MultiPROBE\Robert\Import Data\*.txt" "*.txt"

; ***** Afterwards, disconnect from host!
disconnect
```


Danksagungen

Herrn Prof. Dr. Dieter Oesterhelt möchte ich dafür danken, dass er meine Doktorarbeit vor der Fakultät für Chemie und Pharmazie vertritt und mir damit die Möglichkeit gab, meine Arbeit in der Gruppe „Proteinanalytik“ des Max-Planck-Instituts für Biochemie, Martinsried durchzuführen.

Meinem Betreuer, Herrn Dr. habil. Friedrich Lottspeich möchte ich dafür danken, dass er mir die Möglichkeit gab, mich mit einer hochinteressanten Thematik aus dem Bereich der Bioanalytik zu beschäftigen. Ich danke ihm für sein großes Interesse, seine Anregungen und uneingeschränkte Unterstützung danken. Das mir von Herrn Lottspeich stets entgegengebrachte Vertrauen hat mir immer wieder positive Impulse bei meiner Arbeit gegeben.

Für die großzügige Bereitstellung der im Rahmen dieser Arbeit benötigten Forschungsmittel, insbesondere der hervorragenden instrumentellen Ausrüstung, möchte ich mich bei Herrn Dr. habil. Lottspeich und Herrn Prof. Dr. Oesterhelt als Stellvertreter der Max-Planck-Gesellschaft ausdrücklich bedanken.

Herrn Dr. habil. Christoph Eckerskorn danke ich einerseits für die Erstellung des Zweitgutachtens, andererseits zahlreichen anregenden Diskussionen und Vorschläge im Bezug auf meine Arbeit während seiner Habilitationszeit am MPI für Biochemie.

Bei Herrn Dr. Frank Siedler, dem ehemaligen Leiter des Peptidservice der Abteilung Bioorganische Chemie von Prof. Dr. Luis Moroder, sowie Frau Sigrid Bauer und Hans Stocker vom Peptidservice möchte ich mich für die durchgeführten Peptidsynthesen und die Überlassung zahlreicher Peptide aus dem bestehenden „Fundus“ der Abteilung bedanken.

Herrn Dr. J. Kellermann danke ich für Unterstützung, Rat und Hilfe in fachlicher und organisatorischer Hinsicht.

Bei Frau Bärbel Rinke möchte ich mich für die Durchführung der Aminosäureanalysen bedanken. Herrn Reinhard Mentele danke ich für die Überlassung der „DrTI“-Proben und „DrTI“-Sequenzen. Frau Heidemarie Groß danke ich für die Anfertigung von Gelen und Western-Blots. Allen Kolleginnen und Kollegen danke ich für ihre Hilfsbereitschaft und die freundliche Atmosphäre

Ein ganz besonderer Dank gilt meiner Kollegin Frau Monica Zobawa. Der gemeinsame Erfahrungsaustausch und Kampf an der „MALDI-MS-Front“ war nicht nur ein wichtiger Baustein dieser Arbeit, sondern auch eine persönliche Bereicherung.

Schließlich möchte ich mich bei meinen Eltern bedanken: Für ihre Unterstützung und Hilfe in jeglicher Form während meiner gesamten Studien- und Promotionszeit und für ihre Geduld und ihr Verständnis.

Lebenslauf

Name: Robert Georg Kratzer
Geburtsdatum: 05.12.1969
Geburtsort: Augsburg
Staatsangehörigkeit: deutsch
Eltern: Georg Kratzer
Anna Kratzer, geb. Elster
Familienstand: ledig
Wohnort: Am Eichenwald 2B, 86356 Neusäß

Schulbildung:

1976 – 1980 Grundschole Neusäß „Am Eichenwald“
1980 – 1989 Justus-von-Liebig-Gymnasium Neusäß

Grundwehrdienst:

1989 – 1990 Ulm / Landsberg am Lech

Hochschulausbildung:

Okt. 1990 – Sept. 1996: Studium der Chemie an der Universität Ulm
5. Nov. 1992: Diplom-Vorprüfung
Nov. 1995 Diplom-Prüfung
Dez. 1995 – Sept. 1996: Diplomarbeit bei Prof. Dr. Th. Welsch
9. Sept. 1996: Diplom Chemie
Dez. 1996 – März. 2001: Promotion bei Prof. Dr. D. Osterheld am Max-Planck-Institut für Biochemie, Martinsried, in der Arbeitsgruppe von Dr. habil. F. Lottspeich